PhD-FSTM-2022-032
The Faculty of Sciences, Technology and Medicine

DISSERTATION

Defence held on 22/03/2022 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN Biologie

by

Pedro Miguel TEIXEIRA QUEIRÓS
Born on 22 July1992 in Rio Tinto, (Portugal)

DEVELOPMENT OF DATA INTEGRATION TOOLS WITHIN FUNCTIONAL GENOMICS

Dissertation defence committee

Prof. Dr. Paul Wilmes
Professor and head of the Systems Ecology group, Université du Luxembourg

Dr. Patrick May
Head of the Genome Analysis group, Université du Luxembourg

Prof. Dr. Emma Schymanski, Chairwoman
Professor and head of the Environmental Cheminformatics, Université du Luxembourg

Prof. Dr. Rob Finn
Professor and head of the Microbiome Informatics team, EMBL-EBI

Prof. Dr. Thomas Rattei
Professor and head of the division of Computational Systems Biology, University of Vienna

Affidavit

I hereby confirm that the PhD thesis entitled ”Development of data integration tools within functional genomics” has been written independently and without any other sources than cited.

Pedro Queiros

Digitally signed by Pedro Queiros
DN: C=LU, CN=Pedro Queiros, E=pdqueiros@gmail.com
Reason: I am approving this document with my legally binding signature
Location: Esch-sur-Alzette
Date: 2022.04.02 10:08:20+02'00'
Foxit Reader Version: 10.1.1

Luxembourg, 02/04/2022

Pedro Queirós

1 Preface

My academic background is slightly unconventional; I completed my Nutritional Sciences BSc (which had a heavy focus on Clinical Nutrition) in 2015 and then decided to change into a career that better aligned with my propensity towards the computational fields. I finished my Bioinformatics MSc in 2018, which served as an integral foundation to my programming and Data Science skills. In 2019 I started my PhD, which was a precious learning experience in some highly specific fields in Biology (for which I quickly realised I lacked some knowledge due to my background in Nutrition), in addition, it also allowed me to practice and develop my Data Science skills. Indeed, these past three years allowed me to find a field I'm passionate about - data engineering. I would like to thank the University of Luxembourg for giving me this opportunity. I would also like to thank Patrick May and Paul Wilmes for their mentoring and supervision, and my colleagues and friends for their support. I would like to thank all the jury members and Francesco Delogu and Benoit Kunath for reviewing this thesis. Finally, I would like to thank my family and girlfriend for their unconditional support, you were the supporting pillars that made all of this possible.
In retrospect, this thesis is also the culmination of the many years of my life spent as a student. Having made my fair contribution in the form of publications and past theses, I have allowed myself lighter tones of speech, if only to softly introduce the reader to the heavier and, without a doubt, more serious topic of bioinformatics and derivatives.

Data is not inherently knowledge. This is especially evident in the present Age of Information when data massively towers above the methodologies that attempt to dent it. As with any overwhelming endeavour, it often helps to start by decomposing and analysing the problem by raising different questions: Why and how was this data generated? How is it structured? What do we want to do and what can we actually do with it? While data may be trivially defined as a ”piece of information”, the Science of data cannot be addressed in the same carefree and superficial manner, as data is generated for many reasons, in many ways, and in many shapes; and so, in order to infer knowledge out of data, it is essential to tread mindfully and carefully. This idea of putting good data to good use is the core of this thesis, that is, there is no central biological question or hypothesis, instead, this thesis focuses on how to make the best possible use of the already available towering amounts of data, i.e., data integration. Specifically, this PhD focuses on the integration of biological data which goes in accordance with the interdisciplinary nature of the Systems and Molecular Biomedicine programme of the Doctoral School in Science and Engineering.

2 Abstract

Due to technological advances across all scientific domains, data is generated at an extremely fast pace. This is especially true in biology, where advances in computational and sequencing technologies led to the necessity to develop automated methods for data analysis; thus the field of bioinformatics was born. This thesis focuses on one specific field within bioinformatics - functional genomics. To be precise, in the development of techniques and software for the integration of data to generate novel insights. Indeed, as the amount of knowledge increases, so does the need to integrate it systematically. In this context, the work described herein relates to the integration of multiple resources to improve the functional annotation of proteins, which led to the development of two bioinformatic tools - Mantis and UniFunc. For the downstream integration and analysis of functional predictions, a network annotation tool was developed - UniFuncNet, which, together with the previous tools, enables the efficient functional characterisation of individual organisms or communities.

Index

List of Figures

3 Scientific output

Pedro Queirós, Francesco Delogu, Oskar Hickl, Patrick May and Paul Wilmes, ”Mantis: flexible and consensus-driven genome annotation”, GigaScience, Volume 10, Issue 6, 2021, `https://doi.org/10.1093/gigaScience/giab042`

Pedro Queirós, Polina Novikova, Paul Wilmes and Patrick May. ”Unification of functional annotation descriptions using text mining” Biological Chemistry, Volume 402, Issue 8, 2021 `https://doi.org/10.1515/hsz-2021-0125`

Pedro Queirós, Oskar Hickl, Susana Martínez Arbas, Paul Wilmes and Patrick May, ”UniFuncNet: a flexible network annotation framework”. Manuscript submitted.

Susana Martínez Arbas, Susheel Bhanu Busi, **Pedro Queirós**, Laura De Nies, Malte Herold, Patrick May, Paul Wilmes, Emilie EL Muller, and Shaman Narayanasamy. ”Challenges, Strategies, and Perspectives for Reference-Independent Longitudinal Multi-Omic Microbiome Studies.” Frontiers in Genetics Volume 12, 2021 `https://doi.org/10.3389/fgene.2021.666244`

Tim Van Den Bossche, Benoit J. Kunath, Kay Schallert, Stephanie S. Schäpe, Paul E. Abraham, Jean Armengaud, Magnus Ø. Arntzen, Ariane Bassignani, Dirk Benndorf, Stephan Fuchs, Richard J. Giannone, Timothy J. Griffin, Live H. Hagen, Rashi Halder, Céline Henry, Robert L. Hettich, Robert Heyer, Pratik Jagtap, Nico Jehmlich, Marlene Jensen, Catherine Juste, Manuel Kleiner, Olivier Langella, Theresa Lehmann, Emma Leith, Patrick May, Bart Mesuere, Guylaine Miotello, Samantha L. Peters, Olivier Pible, **Pedro T. Queirós**, Udo Reichl, Bernhard Y. Renard, Henning Schiebenhoefer, Alexander Sczyrba, Alessandro Tanca, Kathrin Trappe, Jean-Pierre Trezzi, Sergio Uzzau, Pieter Verschaffelt, Martin von Bergen, Paul Wilmes, Maximilian Wolf, Lennart Martens, and Thilo Muth, ”Critical Assessment of MetaProteome Investigation (CAMPI): a multi-laboratory comparison of established workflows.” Nature communications Volume 12, Issue 1, 2021 https://doi.org/10.1038/s41467-021-27542-8

Oskar Hickl, **Pedro Queirós**, Paul Wilmes, Patrick May, and Anna Heintz-Buschart. ”binny: an automated binning algorithm to recover high-quality genomes from complex metagenomic datasets.” bioRxiv 2021. https://doi.org/10.1101/2021.12.22.473795

Benoit J. Kunath, Oskar Hickl, **Pedro Queirós**, Camille Martin-Gallausiaux, Laura A. Lebrun, Rashi Halder, Thomas S. Schmidt, Matthew R. Hayward, Dörte Becher, Anna Heintz-Buschart, Carine de Beaufort, Peer Bork, Patrick May, Paul Wilmes. ”Alterations of oral microbiota and impact on the gut microbiome in type 1 diabetes mellitus revealed by multi-omic analysis”. Submitted to the Microbiome Journal in the ”Integrative multi-omics approaches to elucidate microbiome dynamics and ecosystem processes” collection.

Francesco Delogu, Benoit J. Kunath, **Pedro Queirós**, Patrick May, Paul Wilmes ”Can we forecast a microbial ecosystem?”. Manuscript in preparation.

Selected talk in Metaproteomics Symposium 2021 ”Mantis: flexible and consensus-driven genome annotation”

Selected talk in Univeristy of Luxembourg PhD days 2021 ”Mantis: flexible and consensus-driven genome annotation”

Poster presentation in German conference on Bioinformatics 2021 ”Mantis: flexible and consensus-driven genome annotation”

4 Thesis structure

The present document is a cumulative thesis, therefore it is centred around the work developed during my PhD and the respective publications. For this reason, it is organised in such a way that the reader is introduced to the several challenges of biological data integration in respect to the various publications. This thesis starts by discussing the life cycle of data and how several aspects need to be tackled so that data can be successfully integrated. A general introduction follows into various topics (protein function annotation, functional similarity analysis, and network annotation), followed by a brief mention of the data integration challenges inherent to them.

The three first-author publications resulting from this PhD are then presented. This is followed by a general discussion on how the work developed during this PhD addressed these challenges. The publications are as follows:

- Mantis: flexible and consensus-driven genome annotation [176]
- Unification of functional annotation descriptions using text mining [179]
- UniFuncNet: a flexible network annotation framework [174]

In addition, the continued development and use of the tools presented is discussed. Some of the remaining challenges in data integration and future perspectives is also discussed.

5 Introduction

5.1 The life cycle of data

As with many other data-driven fields, Bioinformatics is rooted in the need to use biological data in an automated manner. Bioinformatics, etymologically derived from "*biology*" and "*informatics*", was initially defined in 1970 as the "study of informatic processes in biotic systems" [88]. Bioinformatics started with the application of computational methods for protein sequence analysis [66] but, with advances in molecular biology, computer Science, and sequencing technology [66], bioinformatics soon started using exponentially larger amounts of data (i.e., big data). The advent of big data brought with it new technical challenges in regards to data mining, that is, the extraction of facts, truths, or principles (i.e., knowledge), from the investigation of large datasets.

Data is not inherently knowledge, so researchers are tasked with generating data and transforming it so that new knowledge can be created. This process can be represented as a cycle (Figure 1), where data is generated, processed, and then used to drive new hypotheses and produce new data. The first step in the life cycle of data (Figure 1) is planning. In this step, the case study is designed according to previous knowledge or observations. To this end, past data can be reused (e.g., re-analysis of public datasets) or serve as the foundation for new hypotheses. After planning, the data collection can then start. Because the methods used to generate biological data are highly dependent on the field and case study at hand, the respective representation of this data is therefore naturally inherent to the continuously evolving technical methodologies of the field it originates from (e.g., in the field of genomics, fastq is a common format to store sequencing data). These formats are regularly processed, leading to the branching of the original format into other formats that aim to address more specific tasks (e.g., sequences are extracted from fastq files to create fasta files). However, these data representation formats (either raw or processed) may still not necessarily contain any intrinsic knowledge, so researchers frequently apply further downstream analysis to extract knowledge from this data. This downstream analysis is often not done by the same researchers, while some may generate data, others may be responsible for analysing it. Data and corresponding data-derived knowledge may be integrated in multiple manners, depending on the specific needs of the case study. This leads to the creation of data in various formats which are hard to integrate in an automated and scalable manner. This is not only due to the fact that bioinformatics is a rapidly evolving field, but also because it bridges two very different fields (i.e., biology and informatics), which makes it challenging to decide on ”universal” data formats. Researchers coming from the field of biology often prefer human-readable formats, whereas researchers from the field of informatics tend to prefer machine-readable formats that may be efficiently stored and accessed. Thus, it is crucial to consider both viewpoints’ needs and use formats that address human needs such as readability and interpretability, whilst also preserving the qualities of machine-centric formats (which ultimately provide a crucial quality of data integration - scalability). Data is then stored and later used by the scientific community according to the expertise of each researcher. Hence, data needs to be stored in a manner that allows researchers locally or remotely connected to collaboratively use it. Fortunately, in recent years there have been efforts to standardise data representation and storage, such as the FAIR initiative [236] which intends to provide guidelines to improve the **F**indability, **A**ccessibility, **I**nteroperability, and **R**euse of digital assets. Paired with such guidelines, is the promotion of Open Science [238], which scientific journals, data repositories, and institutions are increasingly adopting [63, 55, 17].

Despite significant efforts, data representation and storage standardisation remain challenging, due to technical, monetary, and subjective reasons (e.g., resistance to adopt guidelines). Data is also often associated with metadata, which, depending on the collection protocols, can be largely unstructured. Additionally, data is often re-analysed and restructured into secondary databases, all of which have their own end-goal and, by extent, structure; for example, the Nucleic Acids Research journal reported, in 2021, a total of 1641 databases just in their database issue [182]. This abundance of data and databases highlights the need to discuss how data can be complementarily used and integrated in a multitude of scenarios.

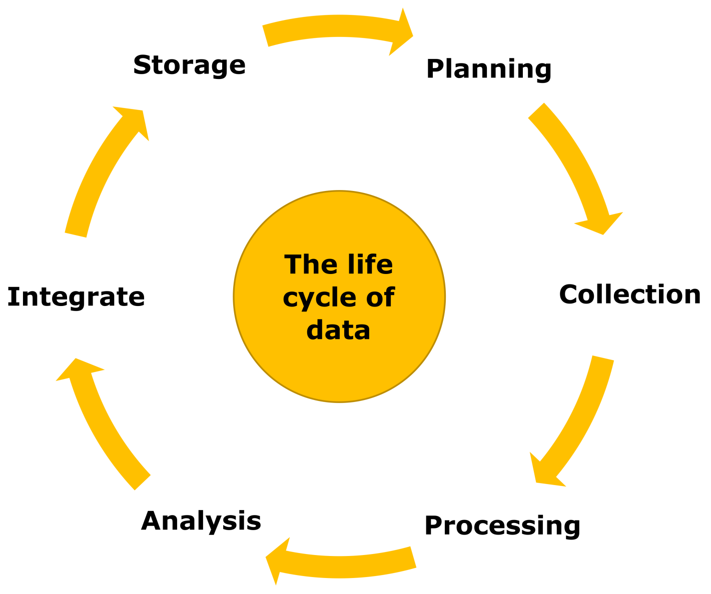


Figure 1 **The life cycle of data** encompasses several integration steps. These range from data-related integration steps to the integration of divergent expertise, and cultural norms. Adhering to collaborative behaviours and guidelines eases the transition between each of these steps.

5.2 Data integration

Due to technological advances, scientific data is routinely generated by thousands of scientists at a quicker pace than ever before [39]. This data is then processed and stored in widely different platforms, leading to the creation of numerous repositories rich in highly complex data. Despite challenges [243, 72, 71] such as: (i) data disparity, (ii) scalability, (iii) accessibility, and (iv) storage and computation costs, data integration (i.e., the process of combining data from multiple sources into one composite resource) is becoming increasingly important. In fact, multiple initiatives and organisations that promote biological data integration have been formed, e.g., ELIXIR [79] and EMBL-EBI [40].

Selecting informative data is challenging, not only due to the existence of copious amounts of highly complex data, but also because of differences (e.g., structure or resolution) in the manner at which this data is generated (for example, due to the end-goal of the case study, or monetary restrictions). At the core of data integration is the identification of shared features between disparate data (either obtained directly or through data transformation); this is followed by the creation of downstream data integration methods that reflect the nature of the respective data, but are also specific to the end-goal of the analysis (which, often lead to the creation of new knowledge bases, i.e., databases). Data integration is not a linear process though, data is often transformed multiple times until it suits the desired framework. For example, enzymatic rates are experimentally measured within a certain experimental context, and this data is published; other researchers then integrate this data into external databases (which for example, requires standardisation of units of measurement); and a researcher working on constraint-based modelling then takes these enzymatic rates and includes them into a metabolic model, which they then archive in a repository such as BiGG [110]. Despite any disparities found in data, data integration is increasingly seen as a necessity in the field of bioinformatics, since using data in a complementary manner tends to lead to a more comprehensive analysis of biosystems [213].
Data selection also entails determining which data should be used based on the premise of providing a solution that scales to the problem at hand. While applying the correct technical methods is essential (e.g., choosing the most appropriate database type and structure), it

is crucial to acknowledge computational bottlenecks, and thus select data accordingly; that is, the amount and complexity of the data used should scale to the scope of the problem at hand. Overall, the selection of data needs to take into account the different aspects of data, whilst also ensuring it can be efficiently used towards the pre-established end-goal.

Accessibility is a more complex issue to tackle since it relies on the community adopting guidelines such as FAIR, which, unfortunately, are still met with some resistance [24, 198]. Accessibility is not only a matter of providing data though, it is also related to, for example, (i) the acquisition cost (which is why it is important to promote open access), (ii) the acquisition ease (e.g., the existence of file transfer protocols), and (iii) the way it is made available (e.g., structured/unstructured). In my experience, a lack of a machine-readable structure is one of the main issues to tackle when trying to integrate biological data. Depending on the data at hand, this lack of structure can even prohibit its use, as the effort to integrate it would outweigh its utility, and the structural variability (i.e., no discernible structural patterns) makes it impossible to parse systematically.

Data also has a monetary and environmental impact that should not be ignored. Indeed, it is progressively relevant to consider the energy consumption associated with data storage [184] and analysis [248, 74]. Therefore, it is vital to promote sustainable data warehousing and the production of efficient data analysis workflows.

To conclude on this topic, the steps preceding data integration can be thought of as a filtering process, that is, data is highly abundant, and so it is the task of the data engineer to select data based on project-specific criteria and requirements. This effectively means analysing multiple datasets (e.g., determining which data is required and how to access it), checking and understanding their structural divergences, efficiently extracting the desired data, and finally being able to develop data models that satisfy the requirements of the resulting composite dataset (Figure 2).

While some of the most general challenges in data integration were covered, this is heavily application-specific; therefore, the various publications enclosed in this thesis discuss in greater detail the various aspects of the data being integrated and the methodology to do so. In order to provide a theoretical background to the reader, the central topics of

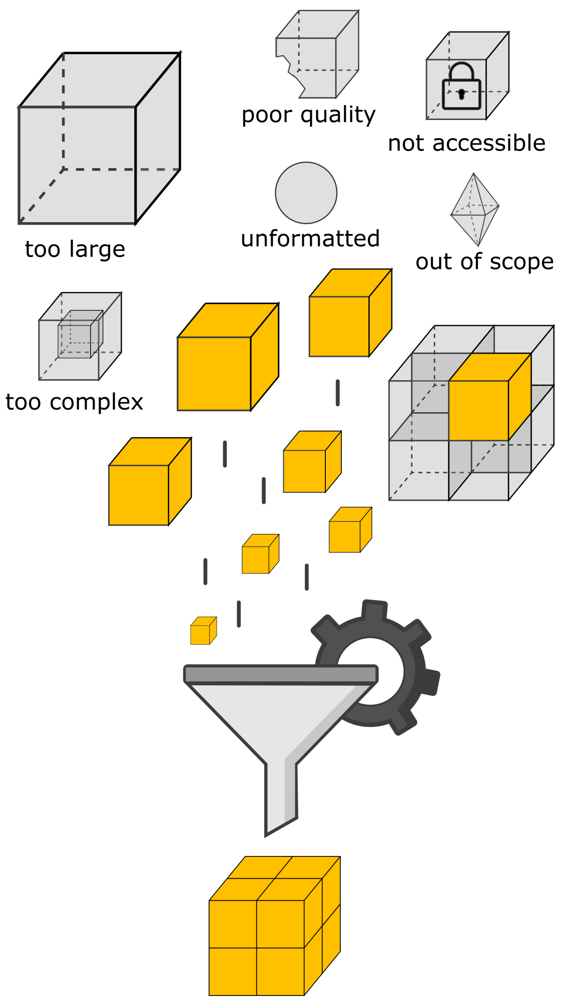


Figure 2 **Data integration** is the process of merging multiple sources of data into a composite dataset. To do so, a thorough analysis and selection of each data source according to the stipulated end-goal/application of the composite dataset are necessary. This entails the selection of data that fits within certain criteria, e.g., the scale, complexity, or resolution of the current application. Data sources may also contain data that is not relevant in the current context, inaccessible, or is not appropriately formatted, being therefore necessary to extract, transform and load the integrated data into a standardised format.

each publication will be introduced as follows: (i) protein function annotation, (ii) function similarity analysis, and (iii) network annotation. Note that each publication provides a more in-depth introduction and thus, each topic will only be summarily introduced in order to avoid redundancy.

5.3 Protein function annotation

Protein function annotation (PFA) is the process of identifying regions of interest in a protein sequence and assigning these regions a certain biological function [176]. PFA has been applied to genomics for many years [58], however, with the arrival of high-throughput shotgun sequencing methodologies it has become increasingly important to rapidly functionally describe the metagenome-assembled genomes (MAGs) [155, 180] recovered from vast and diverse microbial communities. Functional annotation plays an important role in many bioinformatic workflows [153, 247] as it is an integral foundation of many downstream omics analysis (e.g., genomics or proteomics). It may be used, for example, to describe the functional properties of organisms and communities [185, 152], or to create genome-scale metabolic models [18]. Function annotation can be done in several manners, such as:

- sequence homology - where an unknown sequence is compared to a collection of well-annotated sequences (i.e., reference database), and, provided these are sufficiently similar, the function from the well-annotated sequence is transferred to the unknown sequence. Commonly used tools for this purpose include BLAST [3], Diamond [26]) and HMMER [183];
- protein structure similarity - through different methods (e.g., deep learning or crystallography), the structure of an unknown protein sequence is predicted through multiple approaches (e.g., modelling, X-Ray, NMR, Cryo-EM) ; this structure is then compared (using protein structure alignment or 3D structure motif searches) to functionally annotated protein structures, and, provided enough similarity is found, the function is transferred from the functionally annotated protein structures to the predicted structure [234, 122, 103]. Particularly interesting tools for structure prediction and annotation include

AlphaFold [103] and ProFunc [122], respectively;

- genomic context [97] of genes, e.g., using gene neighbourhood; since genes within an operon are functionally associated [112], the function of unknown genes can be infered by the function of their neighbours. Operons are a fairly common gene organisation structure in Bacteria and Archaea (but, to a lesser extent, also present in Eukaryotes [20]) where clusters of genes are co-expressed from a single promoter, i.e., polycistronic transcription [19].

Depending on the omics data being used, PFA can entail functional potential (in the case of genomics) or functional activity (in the case of transcriptomics and proteomics). This is an important caveat to consider when performing functional analysis since functional potential does not directly correlate to functional activity.
Homology relates to the concept of common evolutionary ancestry, which implies that if two sequences share more similarity than what would be expected by chance, then it is more likely that this similarity originates from common ancestry rather than independent evolution [167]. If two sequences are homologs, then it is likely (but not certain) that they have similar structure and function [167]. However, the lack of sequence similarity does not mean that two sequences are not homologs, as there may also be conservation of structure or specific regions within the sequence [167]. Through the use of multiple sequence alignment (MSA) it is also possible to identify conserved regions in sequences (i.e., patterns seen across multiple sequences) and use these to infer function.
As homology is determined by excess similarity, statistical approaches and respective tools are used, such as, Diamond [26] for whole sequence homology or HMMER [183] for region-specific sequence homology. Through the creation of highly comprehensive databases (e.g., eggNOG [94], Pfam [67], and KOfam [6]) it is then possible to identify homologs and transfer the function from the reference database to the unknown sequence. The gold-standard manner to generate reference databases is through experimental validation; however, computationally generated references are becoming increasingly common, where a reference is created via highly stringent computational methods (e.g., eggNOG [95]). Ideally,

computationally-generated reference databases would then be experimentally validated, but this is not feasible at the scale they are created. For ease-of-use, these methods and databases are often wrapped in larger frameworks such as Prokka [194], eggNOG-mapper [96], and RAST [10]. Despite the existence of many of these tools, some issues with the current methods were identified:

- comprehensive integration of region-specific function annotations - typically only the best match (match between an unknown sequence and the reference) is taken into account, which may ignore many of the putative functions of the unknown sequence.
- integration of knowledge from multiple reference sources - it has been shown that including multiple reference databases and using different tools leads to improved functional annotations [75]. Unfortunately, the majority of the tools are associated with either their own or a single reference database, which makes them hard to integrate. If using multiple databases, simply outputting all the possible annotations often leads to redundant or even discrepant functional annotations, which can be problematic in downstream analysis. In addition, different reference databases tend to have different resolutions or use different descriptions and identifiers to describe function, increasing the challenge of integrating them into one composite output. To our knowledge, no tool flexibly integrated multiple databases into a composite non-redundant output.
- lack of taxonomic resolution - many tools do not offer taxa-specific function annotation, which may result in a larger number of wrongly inferred functions due to the use of highly evolutionarily distant ”homologs”.
- setup and customisation issues - many tools have an extremely streamlined implementation, not offering any customisation possibilities. On the other hand, some tools rely heavily on user-expertise, decreasing usability or affecting results quality.
- scalability - many tools scale poorly in a big data context (e.g., poor memory efficiency, no parallelisation, or server-based)

As such, Mantis [176], a protein function annotation tool, was created to addressed these challenges. Regarding the main topic of this thesis - data integration, Mantis is a tool that flexibly (the user can remove or add additional reference databases at will) incorporates multiple databases into a composite consensus-driven output. To do so, Mantis attempts to find matches between unknown sequences and multiple reference databases, it then gathers potential matches, and tries to integrate them by finding a region-specific consensus from the conglomerate of matches. The pre-processing of data (in this context, downloading and structuring the reference databases) is also done automatically, eliminating the need to rely on user expertise. To do so, it was necessary to implement extract, transform, load (ETL) methods so that reference data could be restructured into a standardised format that is modular (which allows for reference databases to be independent of each other) but also fairly straightforward to create by users (which facilitates customisation). In essence, Mantis does not rely on a centralised reference database, instead, reference databases are used independently and then integrated through internal processes. Not relying on a centralised database (which most tools tend to do) eliminates the need to provide regular maintenance, which increases user autonomy (i.e., using Mantis to re-download the reference databases keeps them up to date), but more importantly, it future-proofs Mantis' implementation. Mantis is described in further detail in the respective publication "Mantis: flexible and consensus-driven genome annotation" [176].

5.4 Function similarity analysis

In an effort to integrate functional descriptions from different databases (which enables Mantis to output consensus-driven functional annotations), it was necessary to implement a tool for the similarity analysis of protein functional descriptions - UniFunc [179]. This tool uses natural language processing (NLP) to compare functional descriptions in a pairwise manner.

NLP is the process of exploring and analysing large amounts of unstructured text (in this case, the functional descriptions) through the use of software [61]. While different projects exist in the field of NLP [61], they tend to be specific to the end-goal application. This is because NLP relies on the implementation of methods that take into account the peculiarities of the textual data being used and the application of algorithms that use processed textual data to derive knowledge.

Most NLP workflows tend to start with pre-processing, which entails processing natural/human language/text (i.e., textual data encoding) so that it may be used downstream by different algorithms. NLP can be used for many generic tasks such as:

- part of speech tagging (i.e., lexically classifying words, e.g., ”We are in Luxembourg”, ”are” would be classified as a verb and ”Luxembourg” as a noun);
- named entity recognition (i.e., identify and classify important words within a sentence)
- sentiment analysis (extraction of textual connotation, e.g., expression of subjective features such as happiness or sadness)
- speech recognition

As with any other artificial intelligence field, improper data pre-processing can heavily skew the subsequent data analysis and, by extent, any of the conclusions drawn from it. A large part of NLP encompasses applying methods that can correctly pre-process the data for downstream encoding. While some frameworks contain methods for the pre-processing of data (e.g., NLTK [16], Keras [34], and TensorFlow [138]), it is ill-advised to blindly apply a stack of generic pre-processing methods to a piece of data and expect good results. Instead, the multiple data pre-processing steps should be carefully adjusted or implemented according to the input data. In the context of integrating functional descriptions, data pre-processing can be especially difficult, since these tend to contain highly technical nomenclature that is not easily parsed by conventional tools. Additionally, textual data from multiple databases often contain inter (due to different resolutions) and intra (originating from multiple authors, since no universal scientific writing style exists) differences.

After pre-processing, documents (i.e., a document is an individual entry of textual data) are typically tokenized (i.e., a document is split into tokens - tokens are usually words but may represent more complex entities). These tokenized documents can then be encoded (e.g., transforming tokenized documents into vectors) into a structure that fits the posterior analytical steps. There are many manners to encode tokenized documents, e.g., one hot encoding is one of the simplest. With this method, a zero vector of length C is created, where each vector component represents a token in the vocabulary (i.e., the collection of tokens). If a certain token is present in the document, the component corresponds to 1, if absent to 0. Unfortunately, this method leads to very sparse vectors where all tokens have the same distance between each other, which does not capture semantic similarity. Other methods such as word embedding, encode tokens so that tokens that appear in the same context (i.e., surrounded by similar tokens) are represented by similar vectors, which, by extent, should have similar meanings [78]. In this manner, word embedding methods such as FastText [22] attempt to capture contextual information, which typically leads to better results.

The main objective of pre-processing is thus to provide high quality data that can be encoded into a structure that can be used by posterior methods. The NLP techniques applied for the analysis of the encoded textual data are very diverse and depend on the scope of the problem (e.g., categorisation and summarisation). In the context of UniFunc, the methodology applied enabled the processing of functional descriptions so that a similarity analysis could be done between each encoded functional description. The aforementioned topics and methodologies are discussed in greater detail in the respective publication ”Unification of functional annotation descriptions using text mining” [179].

5.5 Network annotation

Graph theory relates to the mathematical study of graphs, which are structures composed of vertices and edges. In network Science, a graph is commonly referred to as a network, a vertex as a node, and an edge as a link; regardless of the field, the nomenclature is often used interchangeably as the underlying structure is the same. A network is an extremely versatile concept, being therefore capable of capturing many types of entities (e.g., diseases, genes, proteins, and compounds) and having many different biological applications [115], for example, disease maps [60], protein-protein interaction networks [217], metabolic networks [32], and phylogenetic networks [118]. While various graph categories exist (e.g., directed,

hypergraph, weighted), mono and multipartite graphs are fairly common representations of biological data [115] since they can be particularly useful to model omics data where each ome can be typically represented as a subset (”**part**(ite)”) of the respective graph.
VMH’s ReconMap3 visualisation [159] and Metscape [62] (metabolomics data visualisation tool) are good examples of monopartite (i.e., one node type - homogeneous) graphs that represent metabolic networks containing compounds as nodes, and reactions as edges. Gene co-expression networks are also a good example of monopartite graphs where co-expressed genes are linked together and the corresponding network can be used used to predict gene-disease associations [225]. STRING [218] is another example that connects proteins (protein-protein interaction network) through the use of multiple sources such as experimental data, computational predictions, or text mining. Similarly, multiple omics can be integrated through the use of multipartite graphs, where nodes can be partitioned into subsets (according to their node type). For example, Lee *et al* [125] describe how heterogeneous multi-layered networks (HMLN) can be used to integrate omics data in a manner that each layer contains one node type and these nodes are connected through intra or inter-layer edges. An excellent example of a HMLN is Hetionet [87] which contains eleven different types of node types (hendecapartite graph) and multiple types of edges/connections between each node type.
A common practice to integrate biological data within a function-centered network is to predict the functional potential, activity, and interactions within single or multiple layers of omics data [156, 117]. These networks can then be structured according to the end-goal of the analysis, e.g., visualisation [196, 245], identification of key functions [185] or species [152], pathway enrichment and network analysis [120, 85].
In this context, it is possible to use functional annotation information to build multipartite graphs, where each node type (e.g., proteins) can be connected to its corresponding functional annotations (e.g., UniProt ID). Such a network would thus be k+j-partite graphs where k corresponds to the number of subsets containing the main nodes (i.e., the main biological entities such as proteins) and j to the number of subsets containing the sub-nodes (i.e., the type of functional annotation, e.g., UniProt [37] or KEGG [104] IDs). With the use of prior

knowledge, it is then possible to build these functional multipartite graphs, i.e., using reference databases (e.g., KEGG or MetaCyc [30]) to connect functional annotations from some biological entities (e.g., proteins) to others (e.g., reactions). Such a network framework could then be repurposed towards a multitude of downstream analyses. For example, organism-specific networks can be generated by sampling from community-wide networks, creating sub-networks that can used to investigate community interactions [152]. With the end-goal of creating a flexible framework for these types of analysis, a network annotation tool - UniFuncNet [174], was developed. This tool integrates data from multiple databases and allows for the generation of annotated networks. As the goal was to provide a network annotation framework that could be used for multiple purposes, UniFuncNet was implemented so that it could generate annotated networks with four different types of entities (genes, proteins, reactions, and compounds), and also different types of network generation protocols. The integration of data was done through the implementation of several web scraping (i.e., extraction of information from the web) and parsing methods, and posterior merging of data into a composite output. A thorough discussion on the methodologies and applications of UniFuncNet is presented in the respective manuscript ”UniFuncNet: a flexible network annotation framework” [174].

5.6 Scope and objectives of the presented work

To reiterate, this thesis focuses on the development of data integration tools within the domain of functional genomics, that is, the study of how the genome contributes to biological processes. Different Data Science methodologies were applied, and multiple tools were developed to integrate functional annotation data within different contexts.

The first publication relates to the dynamic integration of multiple reference data sources for protein function annotation. Due to the need to use textual data for the integration of multiple sources of data, a function similarity analysis (using NLP) tool was then developed (second publication). Lastly a network annotation tool was developed to provide a holistic approach to functional analysis. This tool enables a comprehensive integration of multiple data sources by using a network-based approach to collect and connect of functional annotation data. Together, these tools enable end-users to perform a highly comprehensive functional analysis of organisms and/or communities.

6 Publications

In this section, all the publication resulting from this PhD will be presented:

- Mantis: flexible and consensus-driven genome annotation [176]
- Unification of functional annotation descriptions using text mining [179]
- UniFuncNet: a flexible network annotation framework [174]

All publications have been adapted to fit with the template of this thesis. In addition, all references and abbreviations have been merged into their respective sections within the present document. Supplementary material, unless specified otherwise, is available at the respective publications doi. All publications are open-access.

6.1 Mantis: flexible and consensus-driven genome annotation

6.1.1 Summary

Protein function annotation can be summarised as the process that identifies regions of interest in a protein sequence and the assignment of a biological function to these regions. Protein function annotation is of interest to numerous downstream analysis, such as functional profiling of an organism or community, describing the functional interactions within a community, and creating genome-scale or community metabolic models. Different methods exist for this process, depending on the available data and the preferred methodology; some of these methods are sequence homology, protein structure, genomic context, and protein networks. This chapter describes Mantis, a protein function annotation tool that attempts to address some specific challenges in this field: (i) region-specific annotation; (ii) integration of multiple reference databases; (iii) integration of taxonomic information; (iv) customisation of reference sources; and (v) scalability.
The authors for this publication[176] are as follows: Pedro Queirós, Francesco Delogu, Oskar Hickl, Patrick May and Paul Wilmes. All authors contributed to the writing, revision, methodology, and study design. I was the main author of this publication and the developer of the tool associated with it.

Mantis: flexible and consensus-driven genome annotation

Abstract

Background: The rapid development of the (meta-)omics fields, has produced an unprecedented amount of high-resolution and high-fidelity data. Through the use of these datasets we can infer the role of previously functionally unannotated proteins from single organisms and consortia. In this context, protein function annotation can be described as the identification of regions of interest (i.e., domains) in protein sequences and the assignment of biological functions. Despite the existence of numerous tools, some challenges remain, specifically in terms of speed, flexibility, and reproducibility. In the big data era, it is also increasingly important to cease limiting our findings to a single reference, coalescing knowledge from different data sources, and thus overcoming some limitations in overly relying on computationally generated data from single sources.
Results: We implemented a protein annotation tool - Mantis, which uses database identifiers intersection and text mining to integrate knowledge from multiple reference data sources into a single consensus-driven output. Mantis is flexible, allowing for the customization of reference data and execution parameters, and is reproducible across different research goals and user environments. We implemented a depth-first search algorithm for domain-specific annotation, which significantly improved annotation performance compared to sequence-wide annotation. The parallelized implementation of Mantis results in short runtimes while also outputting high coverage and high-quality protein function annotations.
Conclusions: Mantis is a protein function annotation tool that produces high-quality consensus-driven protein annotations. It is easy to set up, customize, and use, scaling from single genomes to large metagenomes. Mantis is available under the MIT license available at `https://github.com/PedroMTQ/mantis`.

6.1.2 Background

On a cellular scale, life is, in essence, the activity and the interaction of a plethora of different molecules, among which proteins are responsible for the vast majority of processes. A primary task in understanding how biology works is to resolve its actors properly (e.g., the proteins) and place them into context. The past decades have seen the development of the (meta-)omics fields, unlocking an unprecedented amount of data and deepening our understanding in several fields of biology [195, 151]. Alongside the evolution of the technologies and the increase in data volume, the identification of proteins transitioned from purely experimental techniques (e.g., chemical essays and spectroscopy) toward the computational-based sequence analysis thanks to the discovery of the relationship between conservation of proteins' functions and sequences [235]. Therefore, the current challenges are to make use of the vast number of protein sequences and annotations available and to link new protein sequences to the previously established knowledge. High-throughput methods, such as next-generation sequencing, are able to produce a large amount of data, which then need to be analysed and interpreted. One of the ways to make sense of this data is through protein function annotation (PFA), which is, in the context of this article, the identification of regions of interest (i.e., domains) in a sequence and assignment of biological function(s) to these regions. This strategy has proven effective in the study of single organisms, as well as consortia [8, 33, 99, 83, 139, 166]. Function prediction is based on reference data, i.e., transferring the function from protein X to the unknown protein Y if they are highly similar [235]. Different approaches may be used, the most common being the comparison of an unknown protein sequence to reference data composed of well-studied and functionally annotated proteins (homology-based methods) [216, 229, 23, 208, 194, 96, 10]. Other methods may infer function through the use of machine learning [216, 186], protein networks [244, 218], protein structure [45], or genomics context-based techniques [161], but these are not be covered in this article. For sequence alignment, BLAST [4] or Diamond [26] are commonly used, whereas, for profile hidden Markov models (PHMM), HMMER [183] is most widely used. In PFA, these tools are often integrated into larger pipelines to provide enhanced output interpretability, workflow automation, and parallelization [194, 96, 10, 101]. Some PFA tools target specific taxa [130], while others are designed with large-scale omics analysis in mind [239, 147, 108]; indeed, each PFA tool is designed to cater to its niche research topic. While experimental validation remains the gold standard, PFA, despite its many shortcomings [171], is an increasingly valuable strategy that aims to tackle the progressively more difficult task of making sense of the large quantities of data being continuously generated.

The most common method of processing candidate annotations (i.e., sequences or PHMM that are highly similar to the query sequence) involves capturing only the most significant candidate (”best prediction only”, hereinafter called the **BPO** algorithm). This PFA approach works well for single-domain proteins, but multi-domain proteins may have multiple putative predictions [240, 50, 127], whose location in the sequence may or may not overlap. This selection criterion may potentially lead to missing annotations and is therefore not suitable in complex PFA scenarios. To tackle this problem, domain-specific PFA is necessary. A simple approach, previously discussed in Yeats et al.[240], would be to order the predictions by their significance and iteratively add the most significant one, as long as it does not overlap with the already added predictions (henceforth referred to as the **heuristic** algorithm). Owning to the biased selection of the first prediction, this algorithm does not guarantee an optimal solution (e.g., a protein sequence may have multiple similarly significant predictions). It has been previously shown that incorporating prediction significance and length may produce better results [224]. We implemented a depth-first search (**DFS**) algorithm that improves on the previous approaches.

The selection of reference PHMMs is also critical, because PFA will ultimately be based on the available reference data. Whilst using unspecific PHMMs to annotate a taxonomically classified sample may result in a fair amount of true-positives (TP) results (correct annotations), depending on the confidence threshold used, it may also increase the rate of false-positives (FP) results (over-annotation, due to a less strict confidence threshold) or false-negatives (FN) results (under-annotation, due to a more strict confidence threshold) [193]. Using taxon-specific HMMs (TSHMM) rather than unspecific PHMMs should, in principle, provide better annotations on a taxonomically classified sample, a feature that is already integrated into some PFA tools such as eggNOG-mapper [96] and RAST [10]. In essence, TSHMMs-based annotation limits the available search space, which may have positive and negative consequences. Because the search space is more specific, the annotations produced should be of higher quality; however, this higher specificity of the TSHMM could also lead to under-annotation (incomplete reference TSHMMs) or mis-annotations (low-quality reference TSHMM) [59]. This underlines the necessity to use specific (e.g., TSHMMs) and unspecific PHMMs in a complementary manner. In this regard, the use of multiple sources of reference data remains a challenging aspect of PFA, and, with multiple high-quality reference data sources available, it is increasingly important to coalesce knowledge from different sources. While some PFA tools allow for the use of multiple reference data sources, either as a separate [101] or a unified [96, 7] database, it is still challenging to integrate multiple data sources dynamically.

When using reference data from multiple high-quality sources, the most common and straightforward approach is to consider the output from each reference data source independently (e.g., [101]). However, by doing so, we overlook that many sources can overlap and/or complement each other. Commonly this is compensated for via manual curation, which is feasible only for a limited number of annotations. An automated approach would be to assume only the most significant annotation source for any given sequence and disregard other sources; this may result in vast losses of potentially valid and complementary information (e.g., database identifiers). Because this is not desirable, the challenge is in both in deciding which source(s) provide the best annotation as well as identifying complementary annotations. In the present context, complementary annotations can be defined as functional annotations that are functionally similar but originate from difference data sources; as such, while functionally similar, different data sources are likely to contain information that is absent in other data sources and vice versa. This unique functional information (i.e., database identifiers or functional descriptions) may prove essential in downstream data analysis. A straightforward approach to verify whether functional annotations are functionally similar is to check whether they share a database identifier (ID), e.g.,

(i) Function: ”Responsible for glucose degradation”; IDs: K00844, EC:2.7.1.1, PF03727

(ii) Function: ”Responsible for glucose degradation”; IDs: P52789, PF03727, IPR022673

We can observe that the annotations (i) and (ii) share the database ID PF03727, thus it can be concluded that these annotations are functionally similar. If we were only to select the first annotation, we would ignore potentially useful information (IDs P52789 and IPR022673). However, it may be the case that no IDs are shared between the different annotations, for example:

1. Function: ”Responsible for glucose degradation”; IDs: K00844, EC:2.7.1.1
2. Function: ”Responsible for glucose degradation”; IDs: P52789, IPR022673

We can observe that even though the annotations (i) and (ii) no longer share an ID, they still have the same function ”Responsible for glucose degradation”. Humans can quickly surmise that these annotations are the same because they share the same function description. Should the descriptions be identical or very similar, a machine could achieve the same conclusion with relative ease. However, in our experience, these free-text functional descriptions are often moderately or heavily dissimilar [111, 203], with only a few keywords allowing us to ascertain that they are indeed the same. This then makes it more difficult to use multiple reference data sources. For example:

1. Function: ”Responsible for glucose degradation”; IDs: K00844, EC:2.7.1.1
2. Function: ”Protein is an enzyme and it is responsible for the breakdown of glucose”; IDs: HXK2_HUMAN

In such a scenario, someone trained in a biology-related field can quickly identify the most important words (”degradation”/”breakdown” and ”glucose”) in both sentences and conclude that both annotations point to the same biological function. The challenge is now to enable a machine, deprived of any intellect and intuition, to eliminate confounders (ubiquitous words, e.g., ”the”), identify keywords and their potential synonyms, and reach the same conclusion. A possible strategy is to use text mining, which is the process of exploring and analysing large amounts of unstructured text data aided by software, identifying potential concepts, patterns, topics, keywords, and other attributes in the data [61]. Text mining has been previously used with biological data [231, 170, 242, 202, 92], and even more specifically with regards to gene ontologies [13, 168, 128, 43, 49, 116] and PFA [242]. However, to our knowledge, there is no tool for the dynamic generation of a consensus from multiple protein annotations. This article solves the problem of scaling the integration of different annotation sources, integrating a compact and flexible text mining strategy. We implemented a 2-fold approach to build a consensus annotation, first by checking for any intersecting annotation IDs and second by evaluating how similar the free-text functional descriptions are. This approach attempts to address 3 very relevant issues with PFA [193, 59, 172, 52]: over-annotation, under-annotation, and redundancy. Another challenge in PFA is the lack of flexibility of some tools, as these are often intrinsically connected to their house-generated reference data, and therefore hard to customize. In contrast, we developed a tool that, while offering high-quality unspecific and specific PHMMs, is independent of its reference data, thus being customizable and allowing dynamic integration of new data sources.

We hereby present Mantis, a Python-based PFA tool that overcomes the previously presented issues, producing high-quality annotations with the integration of multiple domains and multiple reference data sources. Mantis automatically downloads and compiles several high-quality reference data sources and efficiently uses the available hardware through parallelized execution. Mantis is independent of any of the default reference data, resulting in a versatile and reproducible tool that overcomes the challenge of high-throughput protein annotation coming from the many genome and metagenome sequencing projects.

6.1.3 Mantis

Mantis is available at `https://github.com/PedroMTQ/mantis`, and its workflow (see Fig. 3) consists of 6 main steps: (i) sample pre-processing, (ii) PHMM-based homology search, (iii) intra-PHMM hits processing, (iv) metadata integration, (v) inter-PHMMs hits processing, and (vi) consensus generation. For future reference, an instance when a PHMM matches with a protein sequence is referred to as a ”hit”. The workflow starts with sample pre-processing, in which the sample(s) is/are split into chunks. This is followed by homology search, where

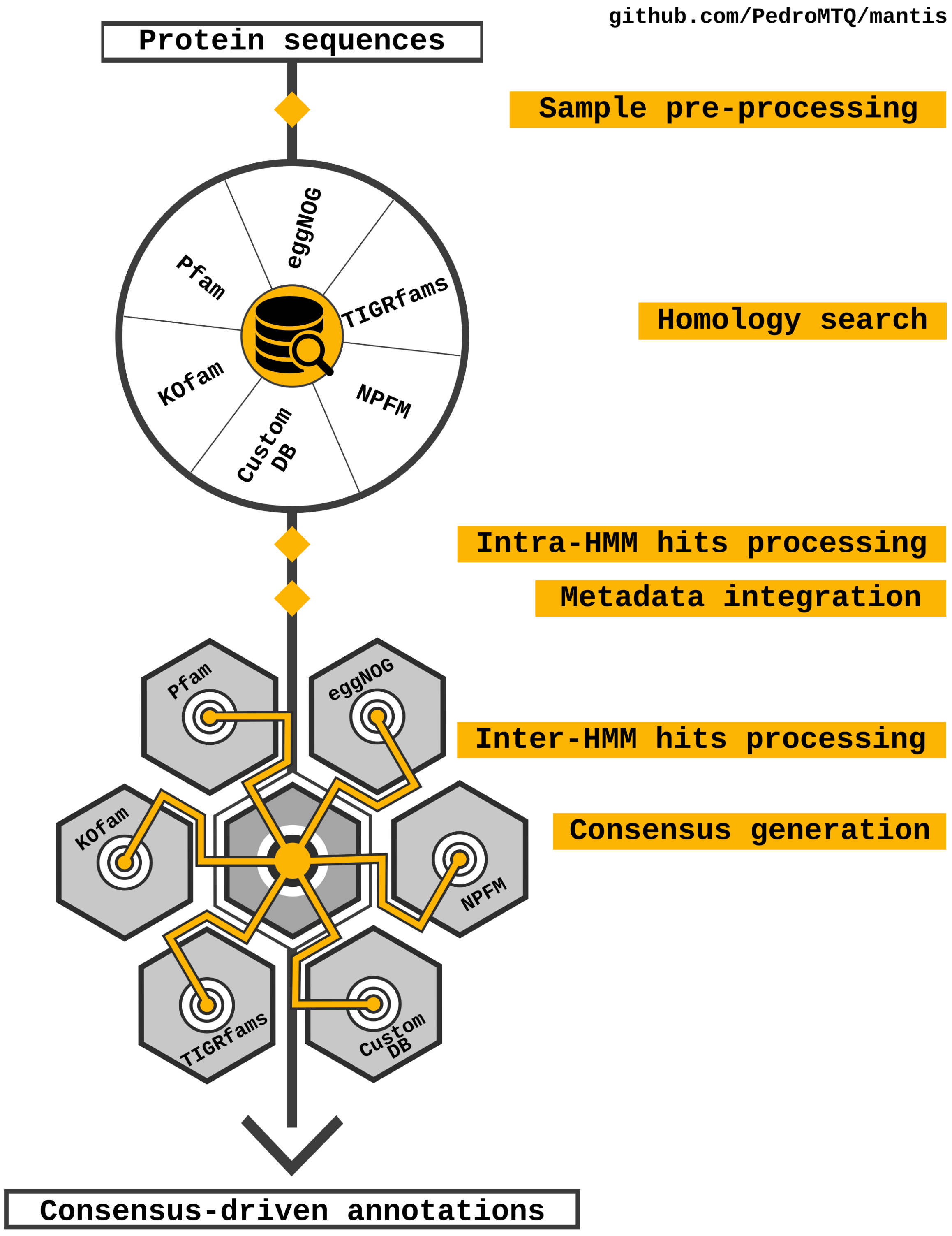


Figure 3 **Overview of the Mantis workflow.** KOfam [6], Pfam [67], eggNOG [95], NCBI protein family models (NPFM) [131], and TIGRfams [76] are the reference PHMMs currently used in Mantis. CustomDB can be any PHMM library provided by the user.

query sequences are searched against the available reference data using HMMER. During intra-PHMM hits processing the DFS algorithm is used to generate and select the best combination of hits per PHMM source; Fig. 4 shows how different algorithms may lead to a different selection of hits. Metadata integration adds the metadata (functional description and IDs) to the respective hits. During inter-PHMMs hits processing, the DFS algorithm is used to generate all the combinations of hits from all PHMM sources (in this step all hits are pooled together). Finally, consensus generation ensures that the best combination of hits among all hits from the multiple reference data sources is selected. This combination is expanded by adding additional hits with consistent metadata (intersecting identifiers or similar functional descriptions) (see Methods section for a detailed description of all these steps). We provide default execution parameters, however, the user is free to fully customize Mantis, not only the parameters but also the reference databases used. Mantis requires a FASTA-formatted protein sequence file as input, where the user can also provide the organism's taxon which will allow for taxa-specific annotation. Reference databases are downloaded automatically. The MANTIS.config file allows for configuration of the reference data and its respective weights and enables the compilation of specific eggNOG TSHMMs. For more details, see the documentation at [175]. Owing to issues with Python's multiprocessing in MacOS, and the fact that HMMER is not available on Windows, Mantis is only available on Linux-based systems.

6.1.4 Analysis

To analyse and validate the performance of Mantis, we performed several *in silico* experiments. We annotated a reference dataset containing curated protein entries from UniProt to set default parameters and evaluate the impact of different Mantis' features: (i) impact of the e-value threshold; (ii) impact of the hit processing algorithm; (iii) how each reference data source contributed to the final output and (iv) impact of the consensus generation on annotation quality. Furthermore, we annotated several sequenced organisms, with and without TSHMMs, thus evaluating the impact of using taxon-resolved reference data. Finally, we compared Mantis against eggNOG-mapper [96] and Prokka [194]. A description of the sam-

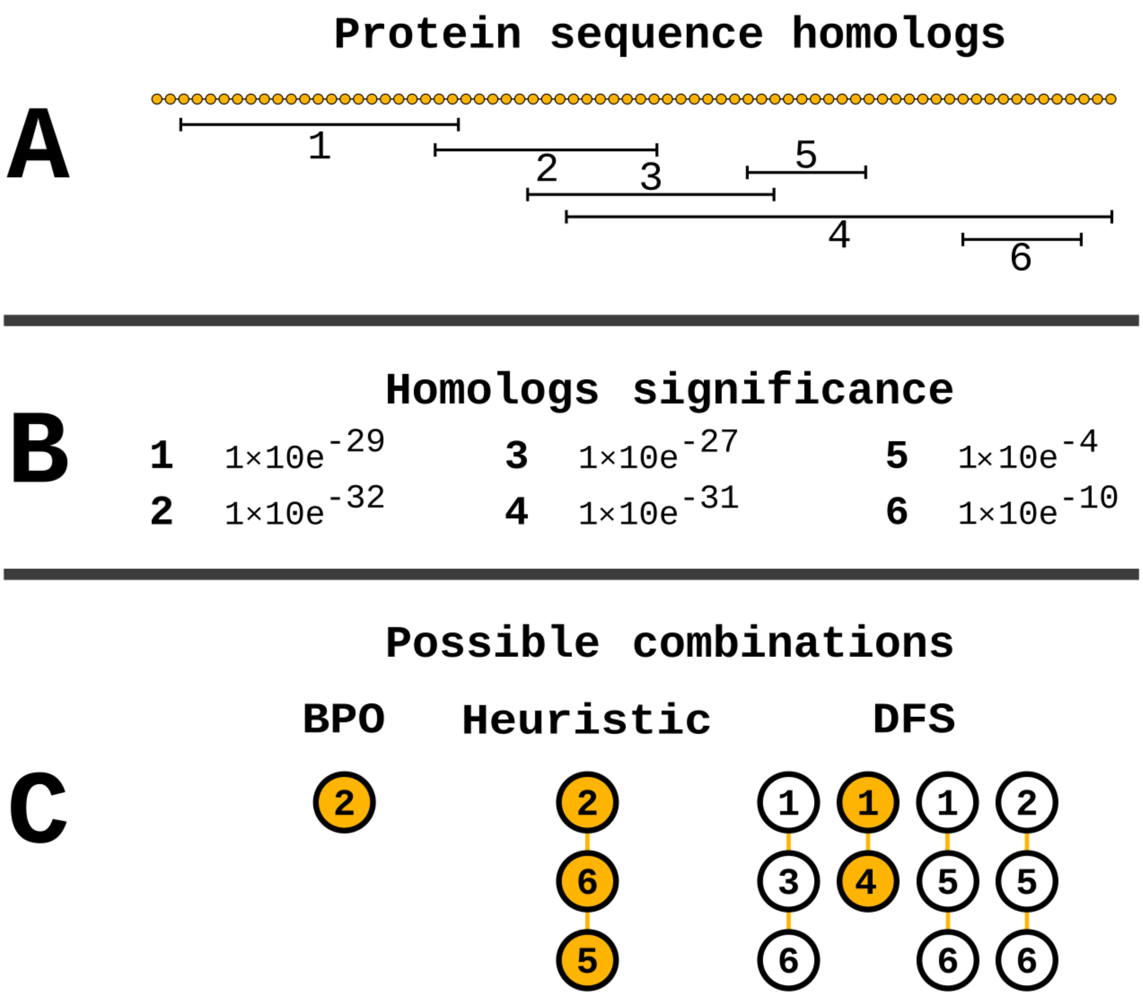


Figure 4 **Homolog selection for the 3 hit processing algorithms in Mantis**. The selection of the hit(s) depends on the underlying algorithm. In the case of the portrayed protein with 6 hits (A) (which are overlapping to various degree) that have varying significance values (B) the three algorithms would behave as follows: (i) BPO would select only the most significant hit (No. 2); (ii) the heuristic algorithm initially selects the most significant hit (No. 2) which then restricts (due to overlapping residues) the hits available for selection (hits No. 1, 3, and 4 can no longer be selected), leading to the selection of the next most significant hit (No. 6), and finally the selection of hit #5; (iii) the DFS algorithm generates all possible combinations of hits, which are then scored according to the e-value, hit coverage and total combination coverage (for more details, please see ”Multiple hits per protein”. According to these parameters, the most likely combinations of hits would be hits No.1 and 4.

ples used for this benchmark is available in ”Sample selection”. Prokka was only used for the annotation of prokaryotic data (i.e., all except for *Saccharomyces cerevisiae* and *Cryptococcus neoformans*). To compare the performance between the different tests, we calculated a confusion matrix for each test. For future reference, a TP occurs when a functional annotation (predicted from a PFA tool) shares one or more database IDs with the respective reference annotation (e.g., Pfam ID); a FP when no database IDs are shared; a FN when the PFA tool does not annotate a protein sequence but a reference annotation is available; and a TN when the PFA tool does not annotate a protein sequence and no reference annotation is available. Precision is defined as $\frac{TP}{TP+FP}$, Recall as $\frac{TP}{TP+FN}$, and F1 score(harmonic mean of precision and recall) as $2 \times \frac{Precision \times Recall}{Precision + Recall}$. The F1 score is used as a performance metric. Further details on the benchmark are available in ”Establishing a test environment”.

Initial quality control

Function assignment e-value threshold It is known that the e-value threshold directly affects annotation quality, however, no gold-standard threshold exists [224]. Depending on the reference data source’s size, quality, and specificity, we may use more or less stringent thresholds. It is therefore essential to test annotation quality with different thresholds. As such, we tested different static e-value thresholds and a dynamic threshold, which has been described in ”Testing different e-value thresholds”. As can be seen in the Supplementary Table 1, precision was similar across the range of e-value thresholds tested, with recall/sensitivity decreasing with lower e-value thresholds. Unexpectedly, unlike recall, precision was not directly correlated with the e-value threshold; indeed a maximum precision of 0.747 was obtained for the e-value threshold $1e^{-6}$, with precision slightly decreasing with more stringent e-value thresholds. A maximum F1 score of 0.827 was observed for the e-value threshold $1e^{-3}$, as such, we chose this value as the default e-value threshold for Mantis.

Impact of hit processing algorithms To understand whether the different hit processing algorithms resulted in statistically significant differences in F1 scores, we created synthetic samples and performed pairwise comparisons between the DFS and the other algorithms: (i) DFS and heuristic, and (ii) DFS and BPO. We rejected the H_0: ”no differences in F1 score between the tested algorithms” in both comparisons since p-value ¡ 0.01. The DFS algorithm resulted in a greater F1 score (mean = 0.827) than the heuristic (mean = 0.826) and BPO (mean = 0.816) algorithms. Further details on results can be found in the Supplementary Table 2, and further details on the testing method can be found in ”Testing hit processing algorithms”.

Impact of sample selection Testing exclusively against well-annotated organisms is a recurring issue with protein annotation benchmarking, resulting in the re-annotation of sequences already present in the reference data used, leading to a biased annotation quality evaluation. To avoid this bias, we downloaded all the curated UniProt (i.e., Swiss-Prot) protein entries (as of 2020/04/14) and selected entries by their creation date such that we have four samples that contain protein entries created in different date ranges (2010-2020, 2015-2020, 2018-2020, and 2020). Samples with more recent protein entries are increasingly more likely to lack any proteins used to generate Mantis' reference data, which increases the likelihood that potential annotations are due to true sequence homology (and not to circular re-annotations). We annotated these samples using three different hit processing algorithms (DFS, heuristic, and BPO), determining the impact of each on the F1 score.
As seen in Fig. 5, the F1 score decreased as the sample was restricted to more recent data. As seen in the Supplementary Table 3, when comparing the hit processing algorithms, we found that the DFS algorithm consistently outperformed the other algorithms, with an mean F1 score 0.021 and 0.003 higher than the BPO and heuristic algorithms, respectively. In addition, the F1 score difference between the multiple hits algorithms (DFS and heuristic) and the single hit algorithm (BPO) increased as the entries in a sample were restricted to more recent years.

Contribution of the different reference data sources We analysed each reference data source's contribution to the output annotation for the UniProt 2010-2020 sample. By

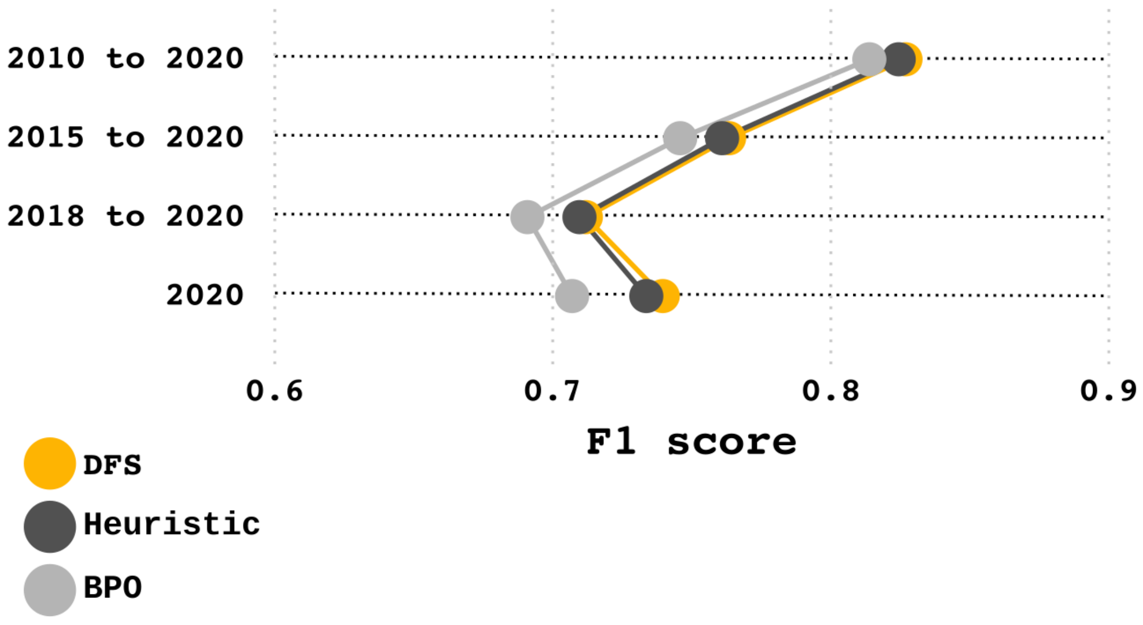


Figure 5 **Annotation F1 score per hit processing algorithm and sample.** Overall, the DFS and heuristic algorithms achieve similar results, outperforming the BPO algorithm.

checking the column "HMM_files" in the $consensus\_annotation.tsv$ file, we found that Pfam was present in 24.4% of the sequence annotations, KOfam in 62.37%, eggNOG in 76.52%, NPFM in 13.91%, and TIGRfam in 12.96%. Note that, since multiple reference data sources may be present in one sequence (due to the consensus generation and hit processing algorithms), the sum of the previous values is above 100%.

Impact of consensus generation During consensus generation, two methods are used for checking the consistency of the hits metadata: IDs intersection and text mining. We analysed the contribution of both methods for the annotation of the UniProt 2010-2020 sample, and found that roughly 35.1% of the consistency checks were due to the text mining approach, and the remaining were due to IDs intersection.

We also tested the impact of text mining on annotation performance: to do so, we annotated the Uniprot 2010-2020 sample but restricted the consensus generation in different manners and with different algorithms. Six different test conditions were created: (i) DFS with default consensus generation, (ii) DFS with consensus generation restricted to IDs (i.e., IDs intersection but no text mining), (iii) DFS without consensus generation (i.e., neither IDs intersection nor text mining), (iv) BPO with default consensus generation, (v) BPO with consensus generation restricted to IDs, and (vi) BPO without consensus generation. We also annotated the same sample using eggNOG-mapper - condition (vii). Prokka was not used here because the present sample contains non-prokaryotic data. The F1 scores were as follows: (i) 0.827, (ii) 0.790, (iii) 0.774, (iv) 0.814, (v) 0.779, and (vi) 0.763, and (vii) 0.703. Further details can be found in Supplementary Table 4.

Hit processing approximation During hit processing, two algorithms may be used, the DFS, and, as a backup (if the DFS algorithm's runtime exceeds 60 seconds), the heuristic. We calculated how many times the heuristic algorithm was used as a backup during the hit processing of the 2010-2020 UniProt sample. We found that for the intra-PHMM hit processing, the heuristic algorithm was used in roughly 7.2% of the sequences, and for the inter-PHMMs hit processing in 0.5% of the sequences.

Quality control with sequenced organisms As a secondary quality control, to assess the impact on F1 score when using taxon-resolved reference data, we annotated several sequenced organisms (for more details, see the Supplementary Table 5) with and without TSHMMs. We also evaluated the impact of the different hit processing algorithms on these samples. As seen in Fig. 6, well-studied organisms (e.g., *Saccharomyces cerevisiae*) had better annotations, especially when applying TSHMMs, unlike poorly described organisms. The mean F1 score gain with TSHMMs was 0.006. With TSHMMs, the DFS algorithm had, on mean, 0.001 and 0.010 higher F1 scores than the heuristic and BPO algorithms, respectively. Without TSHMMs, the DFS algorithm had, on mean, 0.008 and 0.013 higher F1 scores than the heuristic and BPO algorithms, respectively. Further details can be found in the Supplementary Table 6.

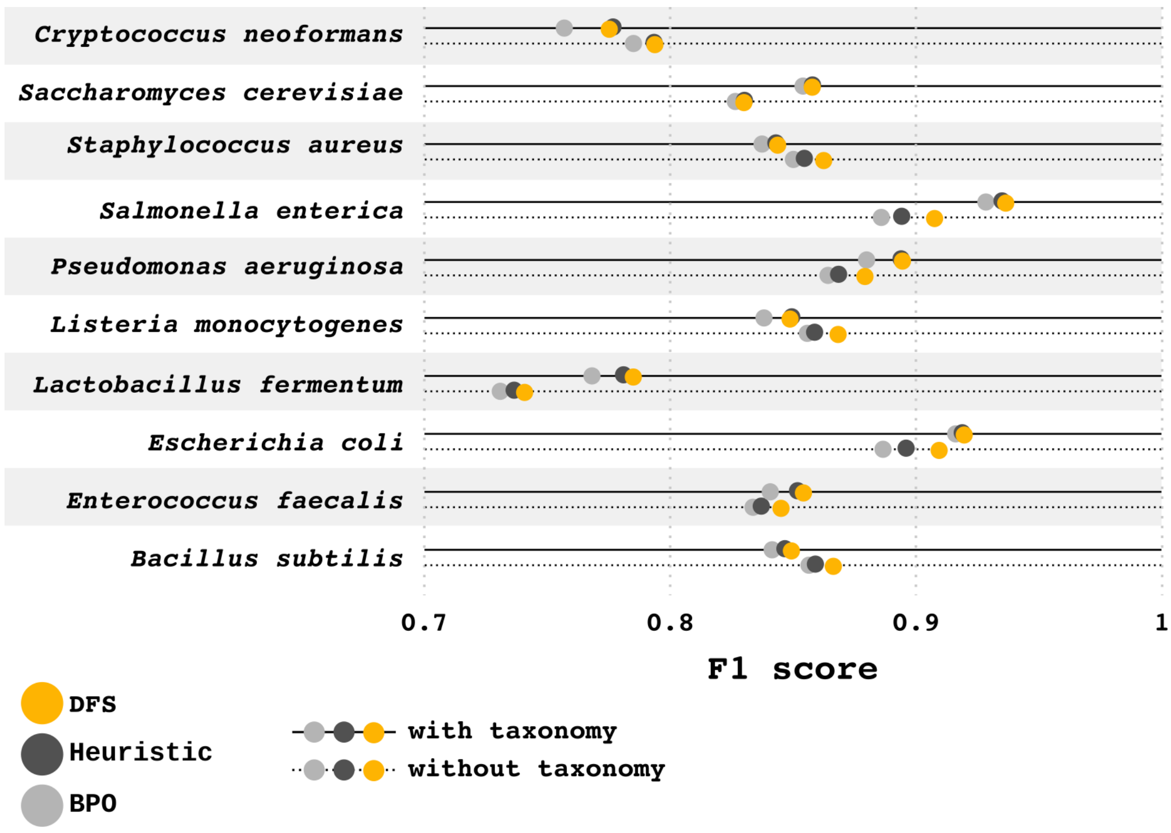


Figure 6 **F1 score per hit processing algorithm and organism, with and without using taxonomy information.** F1 score was higher for well-studied organisms, TSHMMs also tend to perform better with these organisms.

Comparison between Mantis and other PFA tools The sequenced organisms enumerated in the Supplementary Table 5 were annotated with Mantis, eggNOG-mapper, and Prokka (for the latter non-prokaryote organisms were excluded). To evaluate the added value of using the very comprehensive eggNOG reference data source, we also assessed Mantis' F1 score using different reference data. In total, six different tests were performed for each organism: (i) Mantis with default data sources and with taxonomy information; (ii) Mantis with default data sources except for eggNOG's data and with taxonomy information; (iii) Mantis with default data sources but without taxonomy information; (iv) eggNOG-mapper without tax scope option; (v) eggNOG-mapper with tax scope option; (vi) Prokka with default data sources and default execution.
On mean, (i) had a F1 score and annotation coverage of 0.857 and 96.56%, respectively, (ii) 0.832 and 89.82%, (iii) 0.850 and 96.14%, (iv) 0.734 and 88.45%, (v) 0.725 and 88.02%, and (vi) 0.507 and 62.38%. As seen in Fig. 7, Mantis outperformed the other PFA tools in all tests (with one exception in the organism *Saccharomyces cerevisiae*, where eggNOG-mapper without taxonomy had an F1 score of 0.841 and Mantis without taxonomy had an F1 score of 0.830). The mean Mantis F1 score with default execution and TSHMMs was 0.131 higher than eggNOG-mapper (with tax scope) and 0.360 higher than Prokka. Mantis' setting without the eggNOG reference data had an mean F1 score 0.107 higher than eggNOG-mapper (both tools with taxonomy information) and an mean F1 score 0.025 lower than Mantis' with the eggNOG reference data. Further details are available in the Supplementary Table 7.

Annotating metagenomes To our knowledge, there are no manually curated metagenome annotations, therefore annotation validation was not performed, instead we only calculated the annotation coverage. We selected four samples from different environments and predicted the protein coding genes with Prodigal v2.6.3 [98]. The annotated samples were:

- Biogas highly efficient cellulose-degrading consortium (SEM1b) [44, 119] with 39411 sequences;
- Glacier-fed stream sediment (GFS) [28] with 270341 sequences (phenol-chloroform

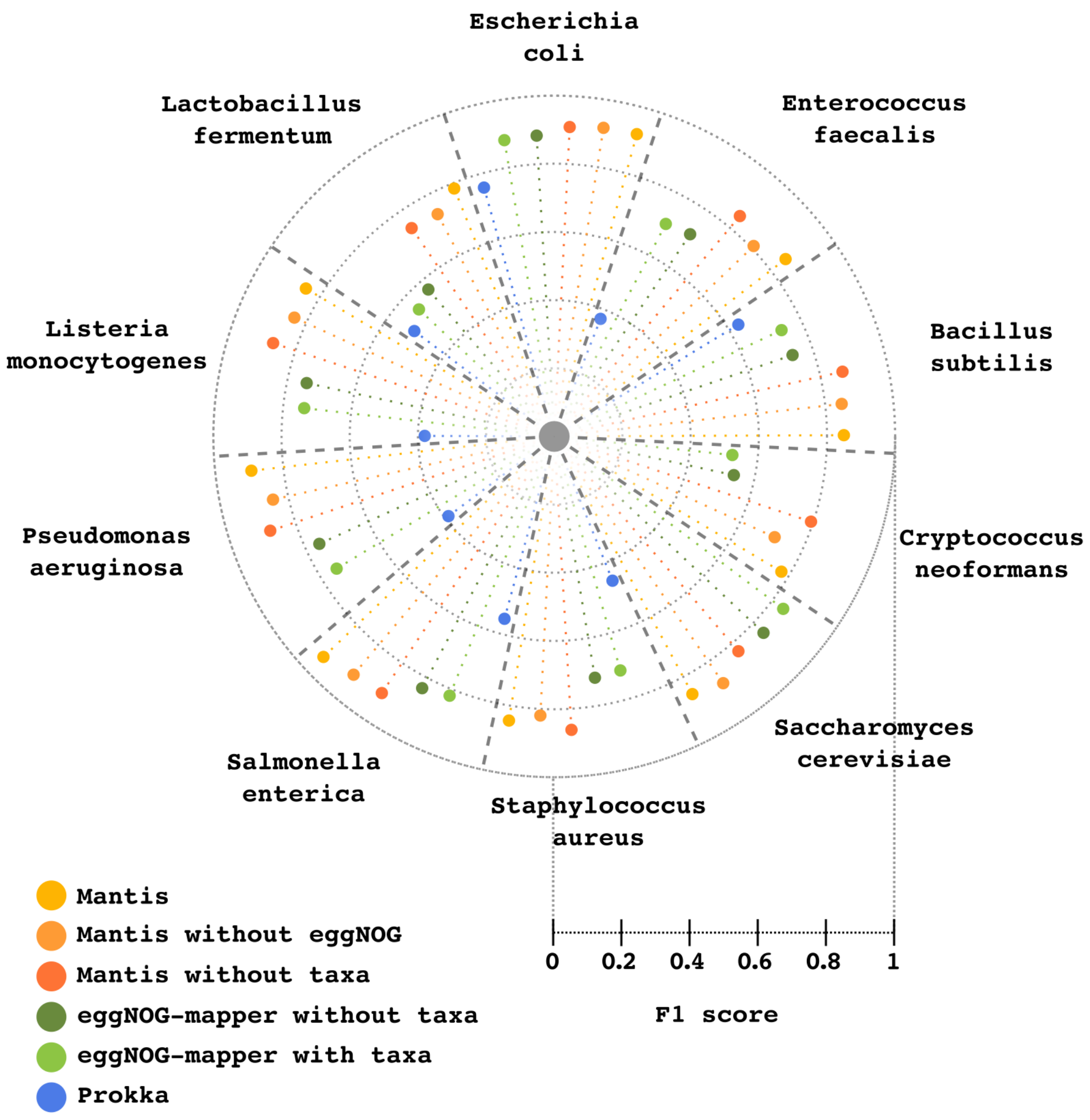


Figure 7 **Annotation F1 score of Mantis, eggNOG-mapper, and Prokka using different reference data.** Each slice represents an organism and contains the F1 score obtained between the different conditions.

extraction batch number 37);

- Marine [214] with 605043 sequences (ERR1726751);
- Human gut microbiome (MuSt [83]) with 692061 sequences (M05-01-V1).

The performance of Mantis varied per metagenome sample; it annotated 213539, 162133, 33016, and 559792 sequences in the samples GFS, marine, SEM1b, and MuSt, respectively. The respective annotation coverage was as follows: 78.99%, 26.80%, 83.77%, and 80.89%. We repeated the same test for eggNOG-mapper and Prokka (in the case of Prokka by annotating the original nucleotide sequences), the coverage for the samples GFS, marine, SEM1b, and MuSt, was, respectively, 77.52% and 10.87%, 16.21% and 1.01%, 81.95% and 32.32%, and 78.72% and 20.37%.

Computational efficiency We ran Mantis against samples with a different number of sequences and a different number of available CPUs. We performed this test for the DFS and heuristic algorithm only. As expected, we found that the heuristic algorithm was faster than the DFS algorithm. The heuristic algorithm was, on mean, 1.42 times faster than the DFS algorithm. As expected, runtimes were inversely correlated to the number of CPUs and sequences. Further details can be found in the Supplementary Table 8.

We also aimed at allowing Mantis to be run on personal computers, which requires removing the eggNOG dataset. However, as we have previously shown in Comparison between Mantis and other PFA tools, this does not cause a high impact on F1 score. We annotated the previously enumerated sequenced organisms (Supplementary Table 5) on a Dell XPS 13-9370 with Ubuntu 20.04.1 LTS 64 bit, 16GB RAM, 512 GB SSD, and an 8 core Intel Core it-8550U CPU. The mean runtime for prokaryotes and eukaryotes was 28 and 93 minutes, respectively. Further details are available in the Supplementary Table 9.

6.1.5 Discussion

We herein presented Mantis, an open-access PFA tool that produces high-quality annotations and is easily installed and integrated into other bioinformatic workflows. Mantis uses a well-established homology-based method and produces high-quality consensus-driven annotations by relying on the synergy between multiple reference data sources and improved hit processing algorithms.

Mantis addresses some major challenges in PFA, such as flexibility, speed, the integration of multiple reference data sources, and use of domain-specific annotations. It also addresses under-annotation through the use of multiple reference data sources, which implicitly leads to a wider search space. Additionally, redundancy, which is a drawback inherent to consensus-driven annotation, is ameliorated by removing duplicate database IDs and/or identical descriptions. We have attempted to avoid over-annotation through the generation of a consensus-driven annotation, which identifies and merges annotations that are consistent (i.e., similar function) with each other (e.g., if three out of five independent sources point towards the same function and two others point towards other, unrelated functions, then these three annotations are more likely to be valid), and eliminating the remaining inconsistent annotations.

We have shown that a stricter/lower e-value threshold did not necessarily lead to a higher F1 score. As expected, a lower threshold restricted the amount of hits, lowering the recall. However, we also found that more stringent e-value thresholds may result in a lower precision; this behaviour is connected to Mantis's consensus generation and hit combinations scoring. A thorough explanation is available in the Supplementary PDF.

Well-curated and commonly used resources were chosen as the default reference data sources for Mantis, containing both unspecific and specific reference data (e.g., taxa-specific). As we have shown, no single reference data source accounted for most annotations, each offering both unique and overlapping insight into protein function, thus confirming their synergy and partial redundancy. These are integrated through a consensus-driven approach, which Mantis uses as an additional quality control step, and a means to automatically incorporate a broader variety of IDs. The intersection of IDs was, as expected, the main contributor towards this integration (since most databases provide cross-linking), however, we found that the text mining approach still contributed considerably (35.12% for the UniProt 2010-2020 sample), which clearly highlights the need to use such a method.

We additionally evaluated the impact of not using text mining during consensus generation and removing the consensus generation altogether on the DFS and BPO algorithms. The benchmark using the BPO algorithm without consensus generation represented the baseline approach towards the integration of multiple reference data sources (merely selecting the most significant hit during inter and intra-PHMMs hit processing). In contrast, the benchmark using the DFS algorithm with the consensus generation depicted the accumulation of all the features introduced by Mantis. Overall, we found a difference of 0.064 in F1 scores, which suggests the additive effect of Mantis's various data integration methods. Mantis, in respect to this specific benchmark, also obtained a F1 score higher than eggNOG-mapper in all conditions, which suggests the importance of using multiple reference data sources.

We have implemented two algorithms for domain-specific homologs search (DFS and heuristic as backup), and have not only shown that these algorithms perform better when annotating previously described protein sequences, but that their impact on the F1 score increased when annotating previously uncharacterized protein sequences (e.g., mean F1 score gain with DFS and BPO algorithms in the UniProt 2010-2020 and 2020 samples was 0.013 and 0.033, respectively). We hypothesize that for the latter, a homology search is not capable of finding whole-sequence homologs, finding, however, multiple domains that partially constitute the protein sequence. As such, we argue that by increasing the resolution (sequence homology to domain homology) of homology-based reference data, domain-specific algorithms may become increasingly valuable. We think this would be especially important when annotating protein sequences without well-described homologs but that contain previously characterized conserved protein domains. In Fig. 8.A, we can observe that the current query sequence is already used to generate the PHMM in the reference data, matching with the PHMM containing it. Such a scenario is common when annotating well-described organisms (e.g., *Escherichia coli*). However, as is often the case when annotating non-model organisms and metagenomes, the query sequence is absent from the reference data (Fig. 8.B), thus partially matching with several PHMMs (which may correspond to multiple domains, depending on the resolution of the reference data). Unlike the BPO algorithm,

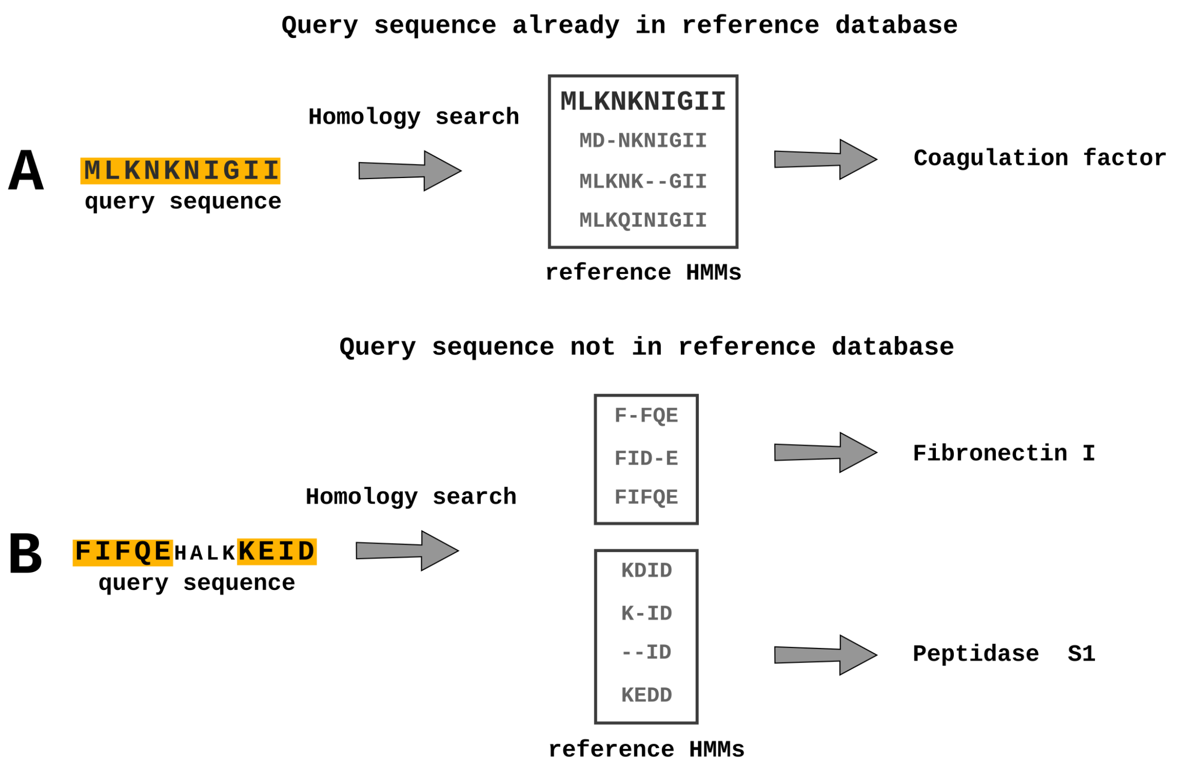


Figure 8 **The impact of the reference data completeness on protein function annotation. A.** the functional prediction is facilitated by the query sequence being previously identified and included in the reference PHMMs. **B.** if the query sequence has not been previously annotated, multiple regions in the protein may match with different reference PHMMs.

the heuristic and DFS algorithms are able to incorporate multiple homologs. While these may not be enough to determine a protein's biological function, they still provide a better biological context than a single functional annotation.

Further improvements in annotation quality may also require the use of motif-based and/or genomic context-based (e.g., operon context information, co-expression, and subsystems) methods such as those described by Sigrist et al.[200], Mooney et al. [149], Mavromatis et al.[140], Overbeek et al.[161], and Hannigan et al.[77]. Nevertheless, the significantly higher F1 score seen when comparing the DFS and BPO algorithms highlights the need to adopt better hit processing methods, especially for non-model organisms. With samples ranging from thousands to millions of protein sequences, sub-optimal hit processing algorithms may cascade into unnoticeable pitfalls in downstream data analysis (e.g., accumulation of incomplete or low-quality genome annotation, which may lead to false biological interpretations). While we have shown that the DFS algorithm outperforms the heuristic algorithm, both achieve a very similar F1 score when applied to non-synthetic samples; since the heuristic algorithm is much more time efficient (as seen in Supplementary Table 8), a user may confidently set it as primary algorithm.

The use of TSHMMs resulted in a 0.006 higher F1 score, however, this improvement (as seen in Fig. 6) was not consistent across all the annotated organisms (as expected, a similar trend was also seen with eggNOG-mapper). We believe this is due to a poorer quality of the TSHMMs for some organisms, which is a consequence of the issues with the current taxonomy classification system [165, 164] and lack of knowledge regarding highly resolved taxa [27]. Model organisms such as *Escherichia coli* and *Saccharomyces cerevisiae* clearly benefited from TSHMMs, both since the reference data already contains data specific to these organisms and that functions of proteins within model organisms are better experimentally described. Conversely, non-model organisms are often only computationally annotated by association, contributing to a weaker reference annotation (which can be observed by the higher rate of potentially new annotations in these organisms, as seen in the Supplementary Table 6). Nonetheless, while experimental evidence remains the gold standard, it is unfeasible to ignore the need for computational methods to infer function. While steps in this direction have been taken [95, 10], taxon-resolved PFA remains a challenge.

We benchmarked Mantis against two other PFA tools - eggNOG-mapper and Prokka, and have shown that Mantis achieves a higher F1 score (0.131 higher than eggNOG-mapper and 0.350 higher than Prokka). Although Mantis' default execution heavily relies on the eggNOG reference data, we have also shown that even without it, it is possible to achieve an almost similar F1 score. This attests to the quality of the various reference data used, showcasing as well the possibility of running Mantis on a personal computer (something that would be impossible with eggNOG's prohibitive size).

We also evaluated the annotation coverage of Mantis and the other PFA tools when annotating metagenomes. Mantis had the highest annotation coverage among the tested PFA tools, but eggNOG-mapper was close behind. All PFA tools had a low annotation coverage for the marine sample. We believe this may be due to a lack of reference PHMMs for this specific environment. This metagenomic sample has data from varying ocean depths, with many novel sequences from viruses, prokaryotes, and picoeukaryotes [215].

Finally, as shown in Accessibility and Scaling, a conda environment and automated reference data download are provided. In addition, Mantis accepts several formats as input (i.e., protein FASTA file, TSV file with paths, directories, or compressed archives), outputting easy to parse TSV files. We believe these features address some of the reproducibility challenges the bioinformatics community still faces [135].

As discussed, there is still room for improvement in the hit processing algorithm DFS (since it does not provide large F1 score gains over the heuristic algorithm). In the future, Mantis could also include genomic context-based annotation methods. Despite the previously discussed challenges, we have clearly shown that Mantis is a flexible tool while also producing annotations with high precision and recall.

6.1.6 Conclusion

By making use of the synergistic nature of differently sourced high-quality reference data, Mantis produces reliable homology-based annotations. By allowing for total customization of these reference data, Mantis is also flexible, easily integrated and adapted towards various research goals. In conclusion, we have shown that Mantis addresses a number of the current PFA challenges, resulting in a highly competitive PFA tool.

6.1.7 Methods

Accessibility and Scaling Mantis automatically sets up its reference data by downloading PHMMs from different sources, and, when necessary, reformatting the data to a standardized format and downloading any relevant metadata. Reference data can be customized via a config file. It also dynamically configures its execution depending on the resources available. A conda environment and extensive documentation [175] are available.

Mantis splits most of the workflow into sub-tasks and subsequently parallelizes them by continuously releasing tasks to workers from a global queue (via Python's multiprocessing module). During each main task of the annotation workflow, workers are recruited (the number of workers depends on the available hardware and work required), these will then execute all the queue tasks. When a worker has finished its job, it will execute another task from the queue, until there are no more tasks to execute. If the queue is well balanced, minimal idle time (time spent waiting for workers to get a new task) can be achieved. Load balancing is achieved by splitting the sample and reference data into chunks. During setup, large reference data sources (more than 5000 PHMM) are split into smaller chunks, this enables parallelization and ensures each annotation sub-task takes approximately the same time. Samples are equally split into chunks (sample chunk size is dynamically calculated). If the sample has 200,000 or fewer sequences, sequences are distributed by their length among the different chunks, so that each chunk has approximately the same number of residues. If the sample has more than 200,000 sequences, then sequences are distributed to each chunk independently of their length (this alternative method is an efficiency safeguard). This two-fold splitting achieves quasi-optimal load balancing. With the sample and reference data in chunks, posterior workflow steps can be parallelized wherever applicable. To note that Mantis uses HMMER's hmmsearch for homology search, which outputs an e-value scaled to the sample/chunk size. Since Mantis splits the samples into chunks, during hit processing, the e-value is scaled to the original sample size.

Input and output MANTIS accepts protein sequence FASTA files as input. If the sample has been previously taxonomically classified, the user can add this information when running Mantis. For example, if annotating an *Escherichia coli* sample, the user could add $-od$ followed by the NCBI ID or the organism name:

```
$ python mantis run_mantis -t sample.faa -od 562
```

Mantis outputs, for each sample, three tab-separated files, each corresponding to a different step in Mantis' workflow: (i) a raw output $output\_annotation.tsv$ (generated during Fig. 3.Intra-PHMM hits processing), with all the hits, their e-value, and coordinates; (ii) $integrated\_annotation.tsv$ (generated during Fig. 3.Metadata integration), with the same information as $output\_annotation.tsv$, but also with hits metadata (e.g., KEGG orthology IDs (KO), enzyme commission (EC) numbers, free-text functional description, etc); and (iii) the main output file $consensus\_annotation.tsv$ (generated during Fig. 3.Consenus generation), with each query protein ID and their respective consensus annotation from the different reference data sources (e.g., Pfam). These files provide contextualized output in a format that is both human and machine-readable. A $Mantis.out$ file is also provided per sample, serving as a log file for each execution step.

Reference data and customization Mantis, by default, uses multiple high-quality reference PHMM sources – Pfam [67], eggNOG [95], NPFM [131], KOfam [6], and TIGRfam [76] (these default PHMMs can be partially or entirely removed). To find more meaningful homologs through taxa-specific annotation, Mantis uses TSHMMs, originally compiled by eggNOG and NPFM. eggNOG TSHMMs were compiled by downloading all the TSHMMs at `http://eggnog5.embl.de/download/latest/per_tax_level/`, their respective metadata originates from the metadata available in the previous link as well as the metadata within the eggNOG-mapper SQL database. NPFM TSHMMs were compiled by downloading all the NPFM PHMMs at `https://ftp.ncbi.nlm.nih.gov/hmm/current/` and assigning each PHMM into their respective TSHMM. A general NPFM PHMM was created by pooling all non-assigned PHMM and the TSHMMS from the following NCBI IDs: 2157 (*Archaea*), 2 (*Bacteria*), 2759 (*Eukaryota*), 10239 (*Viruses*), 28384 (Others), and 12908 (Unclassified). These IDs correspond to NCBI's top level taxonomy rank IDs. A general eggNOG PHMM was created by pooling together the TSHMMs from the same aforementioned NCBI taxon IDs. The user can customize which eggNOG TSHMMs are downloaded by Mantis by adding the line $nog\_tax = NCBI\_ID1, NCBI\_ID2$ to the config file. Custom PHMM sources can also be added by the user, metadata integration of these is also possible (an example is available in Mantis' repository). Since some sources are more specific than others, the user may also customize the weight given to each source during consensus generation. PHMM often only possess an ID respective to the database they were downloaded from, which may not directly provide any discernible information. Mantis, when necessary, ensures that the hits from these PHMMs are linked to their respective metadata. For future reference, while an PHMM is an individual profile, Mantis compiles all related PHMM into a single file making it indexable by HMMER. Thus when a certain PHMM source is mentioned, it refers to the collection of related PHMMs.

Taxa-specific annotation Taxa-specific annotation (TSA) uses the TSHMMs and unspecific PHMM made available by eggNOG and NPFM. TSA, however, works differently from the annotation method of the other reference data. When given taxonomy information (either a taxa name or NCBI ID) the organism's taxonomic lineage is computed (e.g., for *Escherichia coli* the lineage would be `2 - 1224 - 1236 - 91347 - 543 - 561 - 562`). TSA starts by searching for homologs in the most resolved TSHMM (in this case for taxa 562, if it exists). All valid homologs (respecting the e-value threshold) are extracted for each query sequence, and unannotated sequences are compiled into an intermediate FASTA file. A new homology search round starts with the sequences in the current intermediate FASTA, but now in the TSHMM one level above (in this case the TSHMM 561). This cycle repeats until all query sequences have valid homologs or until there are no more TSHMMs to search for. If there are still sequences to annotate, then these homologs are searched for in the general eggNOG and NPFM PHMMs. If no taxonomy information is given, the homology search starts with the general NPFM and eggNOG PHMMs. Non-taxa specific PHMMs (i.e., Pfam, KOfam, and TIGRfams) are always used, regardless of the sample's taxonomy.

Multiple hits per protein HMMER outputs a $domtblout$ file [183], where each line corresponds to a hit/match between the reference data and the query protein sequence. The e-value threshold within the HMMER command limits the amount of hits to be analyzed in the posterior processing steps. Each hit, among other information, contains the coordinates where the query sequences matched with the reference PHMM and the respective confidence score (e-value) (Fig. 4.A and .B). Mantis uses HMMER's independent e-value when using the DFS and heuristic algorithms, whereas it uses the full sequence e-value when using the BPO algorithm (since only the best hit is extracted per protein sequence). For simplicity purposes, both are simply referred to as e-value throughout this paper. The annotation of a protein sequence with multiple hits is a nontrivial problem, thus requiring the implementation of a method for the processing of hits. We designed a method that generates and evaluates all possible combinations of hits by applying the DFS algorithm [106]. This algorithm allows the traversal of a tree-structured search space (i.e., each node is a hit), whilst pruning solutions that do not respect predefined constraints (i.e., overlapping hit residues coordinates), backtracking from leaf to root until the possible solution space is exhausted. Our method generates all the possible combination hits with the following method: (i) Get one hit from the collection of hits and define it as the combination root hit; (ii) Check which other hits overlap up to 10% (default value) [240] with previous hits and select one to add to our current combination of hits; (iii) Repeat step (ii) until no more hits can be added; (iv) Repeat steps (i-iii) so that we loop over all the other hits and all possible combinations are generated. We used Cython [12] to speed up the DFS implementation. Cython is an optimising static compiler for the Python programming language, allowing the compiler to generate C code from Cython code, in this case, functioning as a wrapper for the DFS algorithm. The total number of possible combinations is $2^N - X - 1$, where *N* is the number of hits the protein sequence has, *X* the number of impossible combinations (combinations with overlapping hits), and *1* the empty combination. Due to exponential scaling, this method is not always computationally feasible (e.g., the query sequence is very large and has many small-sized hits). In such a scenario, the DFS algorithm may exceed the system's recursion limit or be unable to find a solution in optimal time (60 seconds by default, but customizable). Should this happen, Mantis employs the previously described heuristic algorithm, which scales linearly (a warning is written in the $Mantis.out$ log).

After generating all the possible combinations, each combination is evaluated according to several parameters:

- $query_{length}$ - number of residues in the query sequence.
- hit_{length} - number of residues in the hit.
- $combo_{length}$ - number of hits in the respective combination.

- *Total coverage* (TC) – number of non-redundant residues in all the combination's hits divided by $query_{length}$. A high TC implies the combination covers a large percentage of the protein sequence.
- *Average hit coverage* (HC) – sum of the coverage of each hit ($\frac{hit_{length}}{query_{length}}$). This sum is then meand by dividing by $combo_{length}$. A high HC implies the hits in the combination are large, thus benefiting combinations with a low amount of large hits rather than combinations with a high amount of small hits.
- *Combination e-value* (CE) – the e-value of each hit is scaled twice, once to reduce the range between different e-values (log10) and the second to understand how each hit e-value compares to the best/lowest hit e-value found for a particular sequence (minmax scaling). The scaled e-values are then summed and divided by $combo_{length}$.

The *combination score* is defined by the following equation:

$$TC \times HC \times CE \tag{1}$$

The combination with the highest *combination score* is then selected, where the available choices will ultimately depend on the algorithm used (Fig. 4.C). Our intra-PHMMs hit processing implementation thus applies a two-fold quality control, initially by limiting the amount of hits in HMMER's $domtblout$ (i.e., e-value threshold) and secondly by hierarchically ordering and selecting the most significant combination of hits.

Using multiple reference data sources An unannotated protein sequence may match with zero, one, or multiple reference PHMM, from one or more data sources. When a protein sequence has multiple hits from different data sources, it is important to identify functionally similar annotations so that no information is lost (i.e., functional descriptions or IDs that may be in one reference data source but not in another). By linking the metadata respective to the PHMM to the now annotated protein sequence, we can identify functionally similar annotations and integrate multiple reference data sources into one final consensus annotation. In this manner, functionally similar annotations are merged, and any complementary information they provide can then be used in downstream analysis (e.g., annotation 1 has a Pfam and KO ID, annotation 2 has an EC number and the same KO ID, merging these will result in a final annotation with more information).

For the integration of functional annotations from multiple data sources, a two-fold approach was used: (i) *Consensus between IDs*; and (ii) *Consensus between the free-text functional description*. The latter is used as a backup, since IDs cross-linking is not universally available. Each reference data source includes metadata relevant to the PHMM herein; this metadata may include multiple intra and/or inter database IDs as well as free-text functional descriptions. IDs are extracted either through source-specific metadata parsing and regular expressions. Free-text functional descriptions are extracted by source-specific metadata parsing. With this information it is then possible to identify annotations that are functionally similar/consistent, and may thus be complementary to each other. The *consensus between IDs* is calculated by identifying intersections between the functional annotations of different reference data sources (e.g., both annotations have the same Pfam ID). IDs within the free-text functional descriptions are extracted (with regular expressions) and also used here. If no consensus between IDs is found, then we proceed with a consensus calculation between functional descriptions (further described in the Supplementary PDF).

Inter-PHMMs hit processing starts by pooling together all hits from the different reference data sources and generating all possible combinations of hits (Fig. 9.A). The same method used in intra-PHMM hit processing is applied, where the DFS algorithm is used by default (again using the heuristic algorithm as a backup), but the BPO and heuristic algorithms can also be used. We then check the metadata consistency (either through IDs or free-text functional descriptions) of each hit against the current sequence's other hits. With this information, a metadata consistency graph is generated (Fig. 9.B). With the metadata consistency graph and all possible combinations of hits, we can then calculate the *consensus combination score* using equation 2. This requires calculating of the *combination score*, using equation 1. This score is then multiplied by an additional score, comprised of the following parameters:

- *Average hit consistency* (HCN) - number of hits (among all hits) with metadata directly consistent (i.e., nodes directly connected in the metadata consistency graph) to the hits in the current combination. Consistency checks are restricted to other reference data sources besides the hit own's reference source (e.g., if a hit is from Pfam, we would only check hits that are not from Pfam). This number, plus the number of hits in the combination, is divided by the total number of hits for the respective query sequence (e.g., if a combination has two hits, with these having metadata consistent with three other hits, and if there are ten hits in total, HCN would equal to $\frac{2+3}{10} = 0.5$). This is an important parameter since it entails independent sources are describing the same function.
- *Reference HMM weight* (HMMW) - mean weight of all the reference data sources within the combination. This is calculated by adding all hits' PHMM weights and dividing this sum by the number of hits in the combination (e.g., if a hit comes from Pfam, that has a weight of 1, and another from eggNOG, that has a weight of 0.8, HMMW would equal to $\frac{1+0.8}{2} = 0.9$). The default weight for each default reference data source has been set according to the authors' perception of the reference quality - creation method, curation level, and annotation completeness (eggNOG - 0.8, Pfam - 0.9, NPFM and KOfam - 0.7, and TIGRfam - 0.5). This weight is customizable, the default weight for custom reference data is 0.7 (which can also be customized).
- *Metadata quality* (MQ) - mean metadata quality of each hit in the combination. If a hit has no annotation data (IDs or description) it is given a score of 0.25, 0.5 if only the description, 0.75 if only the IDs, 1 if IDs and description. All hit's metadata quality score is summed and divided by the number of hits in the combination.

Note that hit metadata consistency (through IDs or descriptions) requires a minimum of 70% residues overlap (default but can be changed). Using the previously calculated *combination score*, we then calculate the *consensus combination score* using the following equation:

$$Combination_{score} \times \frac{HCN + HMMW + MQ}{3} \tag{2}$$

The combination with the highest *consensus combination score* is selected and expanded by concatenating additional metadata from other consistent hits (Fig. 9.C). In this step, consistent hits can be either directly or indirectly connected in the metadata consistency graph (a minimum of 70% residues overlap is still required). This expanded combination is then merged into the final query sequence consensus annotation (Fig. 9.D). Redundant (i.e., repeated identifiers or functional descriptions) or poor quality information (e.g., ”hypothetical protein”) is removed from the consensus annotation.

Sample selection As an initial testing dataset we started by downloading all the curated Uni-Prot [37] (i.e., Swiss-Prot) protein entries created after 2010 (until 2020/04/14), along with their respective sequences, annotations, and annotations scores. We then split these entries by date, 2010-2020, 2015-2020,2018-2020, and 2020 only. For genomic sample benchmarking we selected organisms widely used in microbial community standards. The respective genomes, proteomes, and reference annotations were then downloaded from Uniprot on 2020/05/26 (Supplementary Table 5). These samples were also used for comparing Mantis to eggNOG-mapper and Prokka.

Establishing a test environment For annotation quality benchmarking, we evaluate each annotation produced by Mantis and check whether it agrees (database IDs intersection) with the respective reference annotation, creating a confusion matrix. We created two main types of test samples, the first consisting exclusively of curated UniProt [37] protein entries (and the respective annotations) which were then split by date of creation (2010-2020, 2015-2020, 2018-2020, 2020). The second type consisting of organism-specific UniProt protein entries, with a mix of curated and automatically generated annotations. Each sequence's reference annotation consists of the UniProt protein function annotations. Each sequence reference annotation and the respective PFA tool's annotation is composed of a set of identifiers (if available: enzyme ECs, Gene ontology (GO) IDs, eggNOG IDs, KEGG orthology IDs, Pfam IDs, and TIGRfam IDs) and functional descriptions. During the benchmark process, each sequence's reference annotation (e.g., ”glucose degradation ID1”) is compared

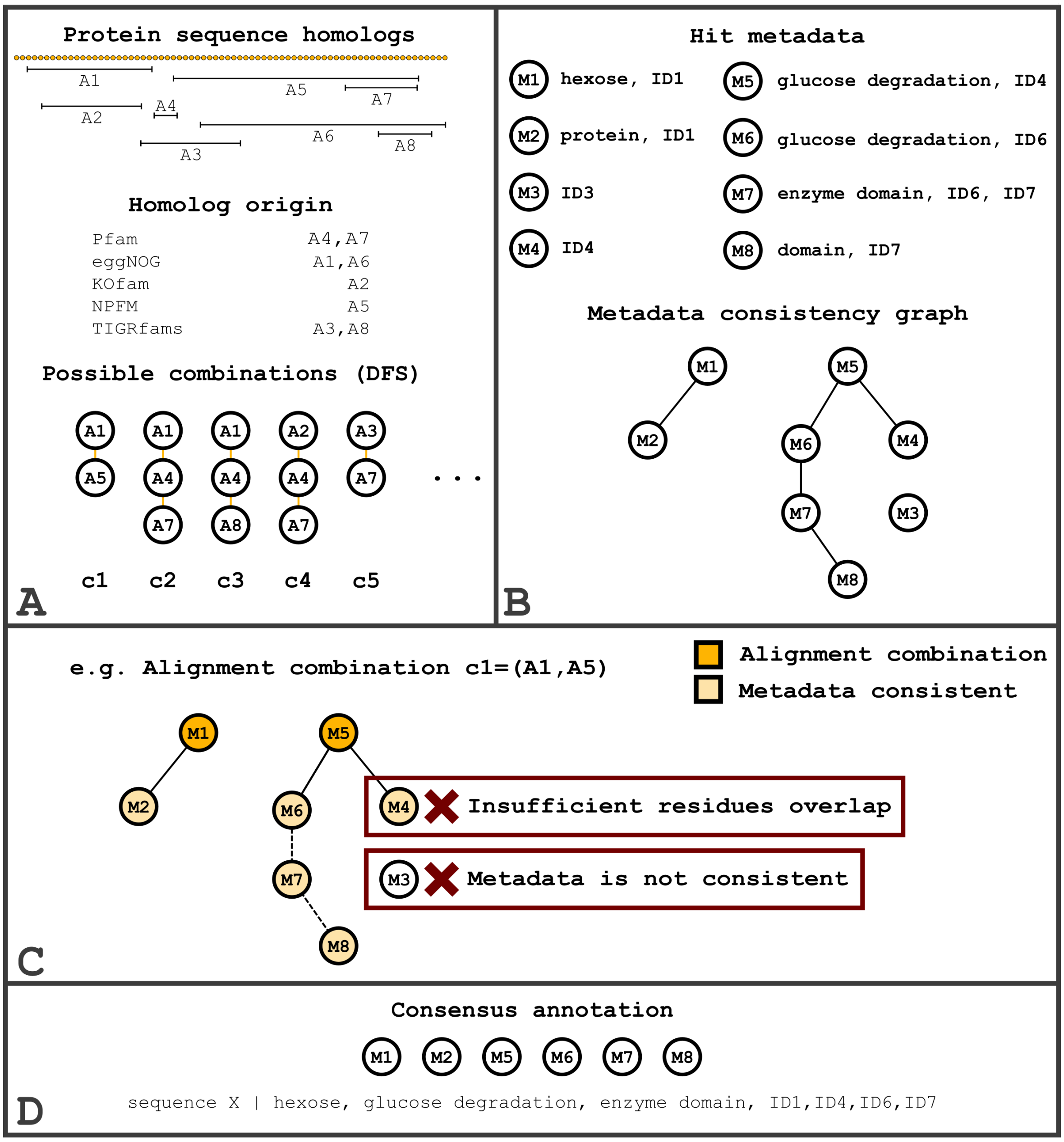

Protein sequence homologs
A1
A5
A7
A2
A4
A6
A8
A3
Homolog origin
Pfam A4,A7
eggNOG A1,A6
KOfam A2
NPFM A5
TIGRfams A3,A8
Possible combinations (DFS)
A1 A5 c1
A1 A4 A7 c2
A1 A4 A8 c3
A2 A4 A7 c4
A3 A7 c5
A
Hit metadata
M1 hexose, ID1
M2 protein, ID1
M3 ID3
M4 ID4
M5 glucose degradation, ID4
M6 glucose degradation, ID6
M7 enzyme domain, ID6, ID7
M8 domain, ID7
Metadata consistency graph
B
e.g. Alignment combination c1=(A1,A5)
Alignment combination
Metadata consistent
Insufficient residues overlap
Metadata is not consistent
C
Consensus annotation
M1 M2 M5 M6 M7 M8
sequence X | hexose, glucose degradation, enzyme domain, ID1,ID4,ID6,ID7
D

Figure 9 **Inter-PHMMs hit processing steps**. Inter-PHMMs hits processing starts by pooling all hits [A1,AN] together (regardless of the reference data source), and generating all the possible (non-overlapping coordinates) combinations [c1,cN] (A). A metadata consistency graph (B) is also built by connecting all nodes [M1,MN] that have intersecting IDs or highly similar descriptions (e.g., A1's metadata M1 is consistent with A2's metadata M2 - shared ID1, and A5's metadata M5 is consistent with A6's metadata M6 - similar description 'glucose degradation'). With this metadata consistency graph, the hit consistency HCN score of each combination is calculated. For c1, for example, a sub-graph containing M1, M5 and all directly connected nodes (only M2 and M6, but not M4, since it has insufficient residues overlap - A4) would be created. The number of nodes in this sub-graph would then be divided by the total number of nodes in the original graph, therefore c1 would have an HCN of $\frac{2+2}{8} = 0.5$. The remaining parameters would then be calculated and the best combination, according to equation 2, would be selected. Finally, if, for example, the best combination is c1, then this combination is expanded by merging all nodes directly or indirectly connected to M1 and M5 in the metadata consistency graph (C) and with sufficient residues overlap (i.e., M2, M6, M7, M8). The expanded combination is then merged into the final consensus annotation (D).

against the PFA tool (i.e., Mantis, eggNOG-mapper, and Prokka) annotation (e.g., ”degrades glucose ID1”). This comparison entails checking whether any of the database IDs present in the reference annotation (i.e., ID1) are also present in the PFA tool annotation (i.e., ID1); if they are, we consider this annotation to be the same. This has some significant limitations: (i) the functional description is the same but the corresponding set of identifiers is not; and (ii) when annotating multiple regions of the protein (which is the case when using Mantis' DFS and heuristic algorithms), it is possible that only one of the annotated regions has IDs that intersect with the respective sequence reference annotation. Unfortunately, due to the different resolutions of the reference PHMMs, it is not always possible to understand whether an annotation refers to a specific domain or a partial whole-sequence hit. While a domain-centric benchmark would be feasible for Pfam, the same is not true for the remaining reference PHMMs with broader resolutions (e.g., TIGRFams provides general functional annotations). However, as we have previously shown, even when using the BPO algorithm, Mantis has shown to output almost equally high F1 scores. Despite these limitations, since whole-sequence reference annotations contain comprehensive cross-linking with other databases, it provides clear benefits: (i) it fits better for the wide-ranging scopes of the reference data sources, and (ii) allows for a more fair benchmark of the different PFA tools that may use different reference data sources (and thus output annotations with different database IDs). This method then allows for the construction of a confusion matrix, where each pairwise whole sequence annotation comparison (PFA tool/reference annotation) corresponds to a single class. TPs occur when the PFA tool generated annotation and the reference annotation share one or more database IDs (e.g., Pfam ID), FPs when no database IDs are shared. FNs when the PFA tool does not annotate a protein sequence, but a reference annotation is available, and TNs when the PFA tool does not annotate a protein sequence, and no reference annotation is available. The functional text descriptions are not taken into account during the benchmark, therefore if an annotation has no IDs, we simply consider there is no annotation. Protein sequences annotated with the descriptions ”unknown function”, ”uncharacterized protein”, ”hypothetical protein” or with Pfam's ”domain-unknown-function”/DUF IDs are not taken into account during benchmarking (for reference and PFA tools annotations). In addition, it is also possible that the reference or PFA tool do not have an annotation for a certain sequence. In any of the these three scenarios, if the PFA tool manages to annotate the sequence, this case is classified as potentially new annotation (PNA). Since no ground-truth exists in these scenarios, PNAs are excluded from the confusion matrix classes (not used during any performance metrics) and are only used to calculate the annotation coverage. PNAs can potentially provide novel insight into protein sequences without any previous annotation. Since, by default, most sequences used during benchmarking will have an annotation, TNs, ergo any metrics using TNs (e.g., specificity), are irrelevant.

Annotation coverage is defined here as the number of annotations produced by the PFA tool divided by the total number of protein sequences in a sample $Total_{seqs}$. $Total_{seqs}$ includes sequences with and without a reference annotation (since not all sequences have a reference annotation), the total number of the PFA tool annotations includes TPs, FPs, and PNAs. Annotation coverage is calculated with $\frac{TP+FP+PNA}{Total_{seqs}}$. Numerous metrics can be calculated from the various confusion matrix categories, we considered precision and recall/sensitivity to be among the most important. Precision is defined as $\frac{TP}{TP+FP}$ and corresponds to the number of correctly annotated protein sequences, out of all the protein sequences the PFA tool managed to annotate. Recall is defined as $\frac{TP}{TP+FN}$ and corresponds to the number of correctly annotated protein sequences, out of all the protein sequences that we know the function of (i.e., protein sequences that have a reference annotation). Both are equally important, a tool with low precision will incorrectly annotate protein sequences, whereas a tool with low recall will not produce sufficient annotations. A way to converge both scores into one is to use the F1 score, which is defined as $2 \times \frac{Precision \times Recall}{Precision + Recall}$. Unless otherwise stated, values shown in this paper are shown as absolute values ranging from 0 to 1.

Finally, we benchmarked Mantis against two other PFA tools - eggNOG-mapper and Prokka. For homology search, Mantis uses HMMER [183], for eggNOG-mapper we used the Diamond-based [26] search (as suggested by the authors), and Prokka uses BLAST and HMMER.

All tests ran on an HPC with Dell C6320, 2 * Intel Xeon E5-2680 v4 @ 2.4 GHz [228], each core had 4GB of RAM. Unless specified, all tests ran with 25 cores and 100GB

RAM (actual Mantis minimum hardware requirements are much lower). In addition, the same methodology and nomenclature apply to any other benchmarked tools described in this paper. Mantis used HMMER v3.2.1. The local version of eggNOG-mapper used was v2.0.6 with database v5.0.1 found at `https://github.com/eggnogdb/eggnog-mapper/commit/41ec3566ab00fd437f905dfde592c553632a9eae`. The local version of Prokka used was v1.14.6 found at `https://github.com/tseemann/prokka/releases/tag/v1.14.6`. For details on execution commands please see the Supplementary PDF.

Testing different e-value thresholds Different e-value thresholds were tested: $1e^{-3}$, $1e^{-6}$, $1e^{-9}$, $1e^{-12}$, $1e^{-15}$, $1e^{-18}$, $1e^{-21}$, $1e^{-24}$, $1e^{-27}$, $1e^{-30}$, and a dynamic threshold. The dynamic threshold was set according to the query sequence length, which was previously shown to provide better results with BLAST [224]. For the dynamic threshold, for sequences with less than 150 amino acids, the e-value threshold was set to $1e^{-10}$, if above 150 and below 250, $1e^{-\frac{sequence_{length}}{10}}$, and if above 250, $1e^{-25}$. The UniProt 2010-2020 sample was then annotated with all the different e-value thresholds, and each output was compared to the reference annotations.

Testing hit processing algorithms In order to understand whether the different hit processing algorithms resulted in statistically significant differences in F1 scores, we created 5000 randomized synthetic samples with 5000 sequences each, which were randomly selected from the 2010-2020 UniProt sample. Per algorithm, we compared the Mantis annotations of each subset to the reference annotations (to allow for pairwise comparison of each algorithm, the same subsets were used in all algorithms). This resulted in a list of confusion matrices (5000 per algorithm), from which we calculated the F1 score. We applied the Wilcoxon signed-rank test, with the H_0: no differences in F1 score between the tested algorithms. As a non-parametric test, this test makes no assumptions on the distribution of the data. A pairwise comparison was done between DFS and the other algorithms: (i) DFS and heuristic, and (ii) DFS and BPO.

6.1.8 Availability of source code and requirements

- Project name: Mantis
- Project home page: `https://github.com/PedroMTQ/mantis`
- Operating system: Linux
- Programming language: Python
- Other requirements: Python 3+, HMMER 3+, and several Python packages (please see the provided environment for a full list)
- License: MIT license at `https://github.com/PedroMTQ/mantis/blob/master/LICENSE`
- RRID: SCR_021001
- Biotools ID: mantis_pfa

6.1.9 Availability of supporting data and materials

The data and code supporting the results of this article are available at `https://git-r3lab.uni.lu/pedro.queiros/mantis_supplements`. The Supplementary pdf ”supplements.pdf” contains: (i) discussion on how the e-value threshold may change Mantis' output, (ii) execution commands, and (iii) information on how the similarity analysis was performed. The *supplements.xlsx* file contains all tables referenced in this article. The first sheet *ToC* contains the table of contents. An archival copy of the code and supporting data is available via the GigaScience repository, GigaDB [178].

Competing Interests The authors declare that they have no competing interests.

Funding Supported by the Luxembourg National Research Fund PRIDE17/11823097.

Acknowledgements Author contributions according to the contributor roles taxonomy CRediT was as follows: Conceptualization: P.Q. and P.M.; Data curation: P.Q.; Formal Analysis: P.Q.; Funding acquisition: P.W. and P.M.; Investigation: P.Q.; Methodology: P.Q. and P.M.; Project administration: P.Q. and P.M.; Resources: P.Q.; Software: P.Q.; Supervision: P.M. and P.W.; Validation: P.Q. (lead), F.D., and O.H.; Visualization: P.Q.; Writing – original draft: P.Q. (lead), and P.M.; Writing – review & editing: P.Q., P.M., F.D. , O.H., and P.W.. All authors proof-read and approved of the content in this research paper.

The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [228]. P.W. acknowledges the European Research Council (ERC-CoG 863664). We would like to thank Tomila Litvishko for proof-reading this research paper. We would like to acknowledge all the creators of the reference data and software used by Mantis, building upon the complementary knowledge of others truly moves the field forward.

In an effort to further understand the impact of the e-value threshold on protein function annotation, I dedicated a supplementary chapter regarding this topic. Below follows the transcript from the supplementary material of the Mantis article.

Impact of the e-value threshold

As an initial quality control of Mantis, we tested different static e-value thresholds and a dynamic threshold to set a default HMMER e-value threshold within Mantis. Interestingly, we saw (supplemental **Table 1**) that a stricter/lower e-value threshold did not necessarily lead to a higher F1 score. By limiting the amount of hits output by HMMER, the e-value threshold will significantly affect the intermediate Mantis' processing steps and therefore its final output.
Naturally, a more stringent e-value threshold results in fewer HMMER hits; this in turn reduces the amount of annotations produced by Mantis, and, by extent, the amount of TPs in the benchmark's confusion matrix.
Unlike TPs, FPs do not necessarily decrease with a lower e-value threshold. For example, the confusion matrix for the sample Uniprot 2010-2020 annotated with an e-value threshold of $1e^{-6}$ and $1e^{-30}$ had 12360 and 12199 FPs, respectively. This goes according to the expectation that a more strict e-value threshold results in better HMMER hits. However, using an e-value threshold of $1e^{-21}$ resulted in 12497 FPs, which contradicts the expected trend (lower threshold would, in theory, equal to less FPs). This occurs because of Mantis' quality control during consensus generation, in particular due to the fact that Mantis attempts to find different reference data sources that point towards the same function. When a higher e-value threshold is used, more hits are available, and thus finding multiple hits that point towards the same function is "easier" than when the available solution space is more limited.
As an example, we selected a protein sequence that has the following reference functional annotation IDs:

```
go:0000160   go:0003677   go:0006355   kegg_ko:K07774   pfam:PF00072   pfam:PF00486
```

Note that while several variables are taken into account for hit combination scoring, for simplicity here we will only refer to the e-value. We then extracted the Mantis functional annotation for the same sequence in the Uniprot 2010-2020 sample, first when using an e-value of $1e^{-6}$ and secondly when using an e-value of $1e^{-21}$. The sample annotated with an e-value threshold of $1e^{-6}$ resulted in the following functional annotations (from the $integrated\_annotation.tsv$ file):

```
Database          HMM hit          e-value  |  Annotation
tigrfam_merged    TIGR01387        2.2e-50  |  tigrfam:TIGR01387
Pfam-A            Response_reg     6.5e-25  |  pfam:PF00072
Pfam-A            Trans_reg_C      5.7e-19  |  pfam:PF00486 description:Transcriptional
                                               regulatory protein, C terminal
NOGG_merged       2SE5K            2.3e-63  |  description:Transcriptional
                                               regulatory protein, C terminal
NCBIG_merged      cztR_silR_copR   2.2e-50  |  tigrfam:TIGR01387
kofam_merged      K02483           2.9e-69  |  cog:COG0745 go:0000156 kegg_ko:K02483
```

For this threshold, the $consensus\_annotation.tsv$ file contained the PHMM hits $2SE5K$ and $Trans\_reg\_C$. The hit with the best e-value was $K02483$, but, since $2SE5K$ and $Trans\_reg\_C$ shared the description "*Transcriptional regulatory protein, C terminal*", these two hits were chosen and merged as the consensus annotation. This consensus annotation shares the ID $PF00486$ with the reference annotation and is therefore marked as a TP.
The same sequence but now using using an e-value threshold of $1e^{-21}$:

```
Database          HMM hit          e-value  |  Annotation
tigrfam_merged    TIGR01387        2.2e-50  |  tigrfam:TIGR01387
Pfam-A            Response_reg     6.5e-25  |  pfam:PF00072
NOGG_merged       2SE5K            2.3e-63  |  description:Transcriptional
                                               regulatory protein, C terminal
NCBIG_merged      cztR_silR_copR   2.2e-50  |  tigrfam:TIGR01387
kofam_merged      K02483           2.9e-69  |  cog:COG0745 go:0000156 kegg_ko:K02483
```

In this case, the hit $Trans\_reg\_C$ is no longer available since it had an e-value of 5.7e-19, which is above the e-value threshold. Consequently, the $consensus\_annotation.tsv$ file contained instead the PHMM hit $K02483$, since, among all available hits, it's the one with the lowest e-value. The consensus annotation now does not share an ID with the reference annotation and is therefore marked as a FP.
The e-value threshold can also have an impact on the choice of the best combination. Since

the combination e-value is calculated by scaling with log10 and minmax, different e-value thresholds will result in different minmax values, and, by extent, different scores for the same combination of hits. For example, if we find 3 hits when using an e-value threshold of $1e^{-3}$:

- hit 1 with e-value 1e-5
- hit 2 with e-value 1e-15
- hit 3 with e-value 1e-10

By applying log10 and minmax scale to each hit we get:

- hit 1 with minmax log10 e-value of 0
- hit 2 with minmax log10 e-value of 1
- hit 3 with minmax log10 e-value of 0.5

Now with an e-value threshold of $1e^{-6}$:

- hit 2 with e-value 1e-15
- hit 3 with e-value 1e-10

By applying log10 and minmax scale to each hit we get:

- hit 2 with minmax log10 e-value of 1
- hit 3 with minmax log10 e-value of 0

In the first scenario, should hit 3 score well in the other ***combination score*** variables, it can still be picked above hit 2 (should hit 2 score poorly in the other ***combination score*** variables). In the second scenario, it is highly unlikely that hit 3 will be chosen since it is now has the worst e-value of all the hits.

While these are anecdotal examples, they depict why and how different e-evalue threshold may lead to unexpected results.

6.1.10 Concluding remarks

The published version of this paper can be found at `https://doi.org/10.1093/gigaScience/giab042`

The GitHub repository for this tool is available at `https://github.com/PedroMTQ/mantis`

In this manuscript, the Mantis protein function annotation tool was described and how it addresses the previously enumerated challenges. Extensive benchmarking was also performed with multiple samples, environments, search parameters and reference data. Finally, Mantis' overall annotation quality was compared against other state-of-the-art tools.

In order to integrate multiple reference sources, it was necessary to develop a NLP tool that compares protein function annotations from multiple sources, being thus generalizable to different writing styles but also being specific to the particularities of protein function descriptions (e.g., highly specific nomenclature).

6.2 Unification of functional annotation descriptions using text mining

6.2.1 Summary

Functional descriptions often contain discrepancies (e.g., different nomenclature) between and within databases, which required the development of a specialised method for the correct pre-processing and similarity analysis of functional descriptions. UniFunc was therefore developed to address this issue and then integrated into Mantis to allow the use of multiple reference sources during protein function annotation.
The authors for this publication[179] are as follows: Pedro Queirós, Polina Novikova, Paul Wilmes and Patrick May. All authors contributed to the writing, revision, and study design. I was the main author of this publication and the developer of the tool associated with it.

Unification of functional annotation descriptions using text mining

6.2.2 Abstract

A common approach to genome annotation involves the use of homology-based tools for the prediction of the functional role of proteins. The quality of functional annotations is dependent on the reference data used, as such, choosing the appropriate sources is crucial. Unfortunately, no single reference data source can be universally considered the gold standard, thus using multiple references could potentially increase annotation quality and coverage. However, this comes with challenges, particularly due to the introduction of redundant and exclusive annotations. Through text mining it is possible to identify highly similar functional descriptions, thus strengthening the confidence of the final protein functional annotation and providing a redundancy-free output. Here we present UniFunc, a text mining approach that is able to detect similar functional descriptions with high precision. UniFunc was built as a small module and can be independently used or integrated into protein function annotation pipelines. By removing the need to individually analyse and compare annotation results, UniFunc streamlines the complementary use of multiple reference datasets.

6.2.3 Introduction

Protein function annotation is the process of identifying regions of interest in a protein sequence and assigning a certain biological function to these regions. It enables the understanding of the physiology and role of single or multiple organisms in a community/ecosystem, which is particularly important to define the metabolic capacities of newly sequenced organisms or communities [204]. Function assignment is based on reference data, such that if we have an unknown protein X which is similar to protein Y (e.g., by sequence or structure similarity) then we can infer these proteins share the same function(s) [129, 235]. By extent, we can then assume that protein X can be assigned the same protein functional description and/or database identifiers (ID) such as protein Y.

The reference data used for this purpose usually stems from a single source, however, some protein function annotation tools use multiple sources, e.g., InterProScan [102]. The latter provides several advantages: reinforcement of annotation confidence (several independent sources indicating the same function), improvement of downstream data integration (wider variety of different database IDs), and higher annotation coverage (wider search space). Simultaneously, using multiple references gives rise to various challenges, in particular, uncertainty to define which annotation best represents the functional role of a certain protein (especially when multiple sources infer mutually exclusive functions) and dealing with the introduction of redundancy. These disadvantages are commonly addressed via manual curation, however, this is not feasible in large-scale projects. Many databases provide cross-linking (e.g., UniProt [37]), which permits automating this process by checking for intersecting IDs, however, some functional annotations only contain free-text descriptions. Applying the same intersection methodology for text is not viable due to human language's intrinsic richness in confounders (e.g., determiner ”the”). This leads to the omission of such functional annotations and may cause the exclusion of potentially useful information. Through the use of text mining techniques, it becomes possible to integrate these functional annotations.

Text mining is the process of exploring and analysing large amounts of unstructured text data aided by software. It allows identifying potential concepts, patterns, topics, keywords, and other attributes in data [61]. A review by Zeng et al. [242] has shown some of the

potential and diverse techniques and applications of text mining in bioinformatics, some of which include literature mining [218, 231], protein research [230], and ontologies [202].
We herein present **UniFunc** (Unified Functional annotations), a text mining tool designed to assess the similarity between functional descriptions, thus allowing for the high-throughput integration of data from multiple annotation sources. UniFunc is available at `https://github.com/PedroMTQ/UniFunc`

6.2.4 Results

UniFunc UniFunc's workflow is composed of four main steps: (i) pre-processing, (ii) part-of-speech tagging (PoST), (iii) token encoding and scoring (TES), and (iv) similarity analysis. The first two steps comprise the natural language processing (NLP) of the functional descriptions (e.g., removing extra spaces and eliminating confounders), the last two the text mining (i.e., text encoding and similarity analysis). Figure 10 showcases UniFunc's workflow when comparing two functional descriptions, each step is described in ”Material and methods”.
While context dependant, we will henceforth refer to an annotation as a functional description of a particular protein (e.g., ”glucose degradation”). Each annotation may contain one or multiple sentences, which are composed of one or multiple tokens (e.g., ”glucose”). A collection of independent annotations constitutes here the annotation corpus.

Benchmark As validation, we downloaded all of Swiss-Prot's [37] protein entries (N=563973, as of 2021/01/16) and selected those that had a functional description, and at least one EC number, Pfam [67] ID, or eggNOG [94] ID resulting in a validation dataset with 133450 entries. We then generated two sets of pairwise functional annotation comparisons. One set of pairs with intersecting identifiers (positive cases) and the other with non-intersecting identifiers (negative case). We then calculated the similarity score of the functional descriptions in each pairwise comparison using different models. In order to understand the impact of the NLP and text mining approach used in UniFunc we built a baseline model that employs very simplistic pre-processing and text encoding methods. We also compared UniFunc against a SciSpacy [154] model, a python library that offers biomedical NLP models. With this pre-

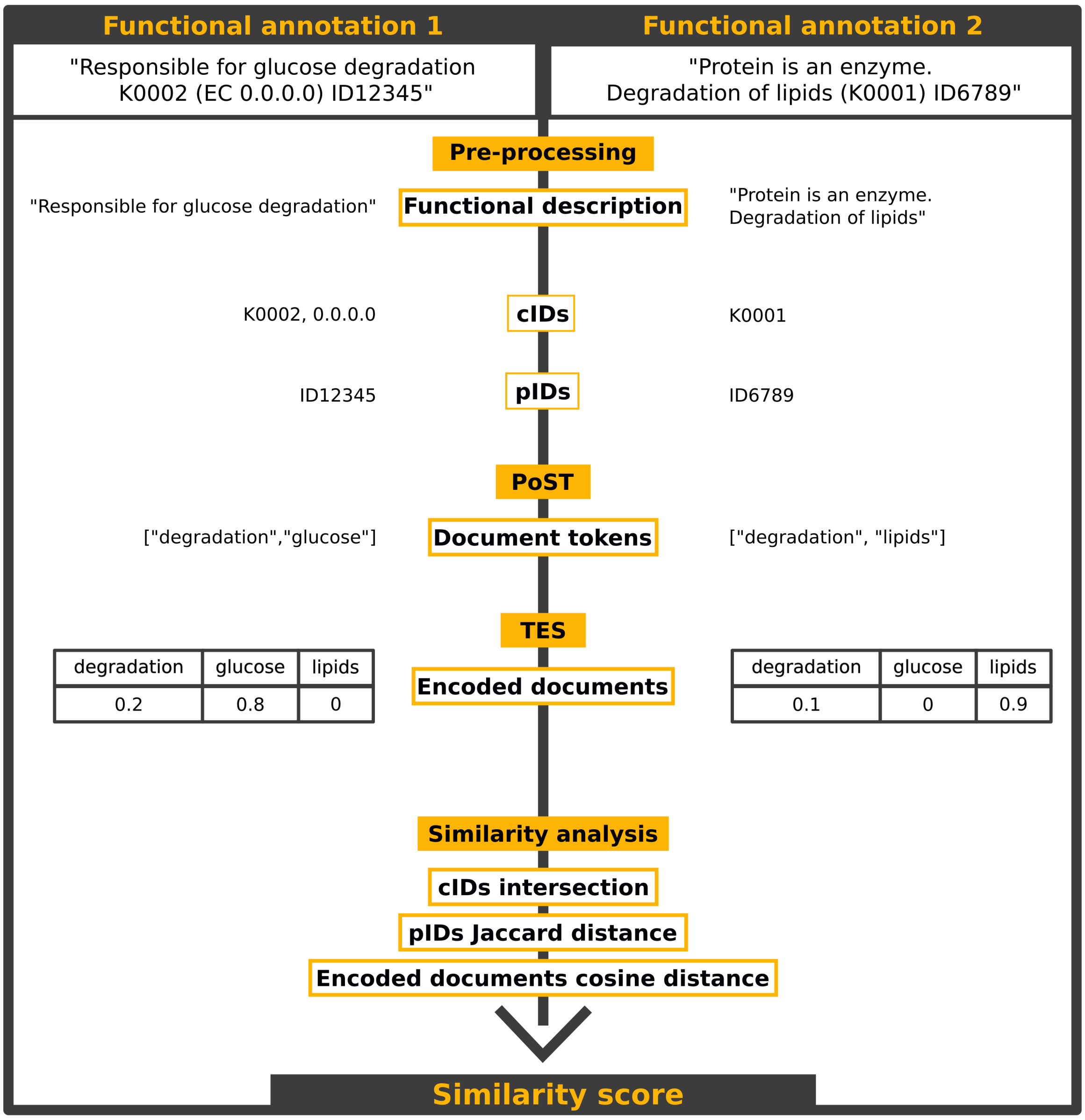

Functional annotation 1
"Responsible for glucose degradation K0002 (EC 0.0.0.0) ID12345"
Functional annotation 2
"Protein is an enzyme. Degradation of lipids (K0001) ID6789"
Pre-processing
"Responsible for glucose degradation"
Functional description
"Protein is an enzyme. Degradation of lipids"
K0002, 0.0.0.0
cIDs
K0001
ID12345
pIDs
ID6789
PoST
["degradation","glucose"]
Document tokens
["degradation", "lipids"]
TES
degradation glucose lipids
0.2 0.8 0
Encoded documents
degradation glucose lipids
0.1 0 0.9
Similarity analysis
cIDs intersection
pIDs Jaccard distance
Encoded documents cosine distance
Similarity score

Figure 10 **Overview of the UniFunc workflow**. UniFunc starts by extracting all the IDs (cIDs and pIDs). It then removes uninformative parts from the annotation and splits it by sentences and tokens. UniFunc then furthers processes the tokens, by tagging each token, and only keeping the most relevant tokens within each annotation and these tokens are encoded into TF-IDF scaled vectors. Finally, the cosine distance between the two annotation vectors and the Jaccard distance between the pIDs are calculated. If any cIDs intersected, the similarity score is 1, otherwise, both previously mentioned distances are used to calculate the entry's similarity score. Abbreviations used in this figure include cIDs (common database identifiers), pIDs (possible database identifiers), PoST (part-of-speech tagging), and TES (token encoding and scoring).

trained SciSpacy model we used the same simplistic pre-processing as the baseline model. All three models used the same similarity metric, i.e., cosine distance.

The results of this pairwise similarity comparison were then compiled into threshold-specific confusion matrices, where true positives (TP) correspond to pairs of entries with intersecting identifiers and a similarity score above the threshold, true negatives (TN) to non-intersecting identifiers and a similarity score below the threshold, false positives (FP) to non-intersecting identifiers and a similarity score above the threshold, and false negatives (FN) to intersecting identifiers and a similarity score below the threshold. These matrices were then used to calculate several performance metrics: $Specificity = \frac{TN}{TN+FP}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F\beta score = \frac{(1+\beta^2)\times Precision \times Recall}{\beta^2 \times Precision + Recall}$ with β equal to 1. These metrics are available in the supplemental excel spreadsheet. We then plotted precision and recall (Figure 11) with a threshold ranging from 0 to 1 in increments of 0.01. The area under the ROC curve (AUC) was also calculated, with the baseline model having an AUC of 0.835, UniFunc 0.868, and SciSpacy 0.876.

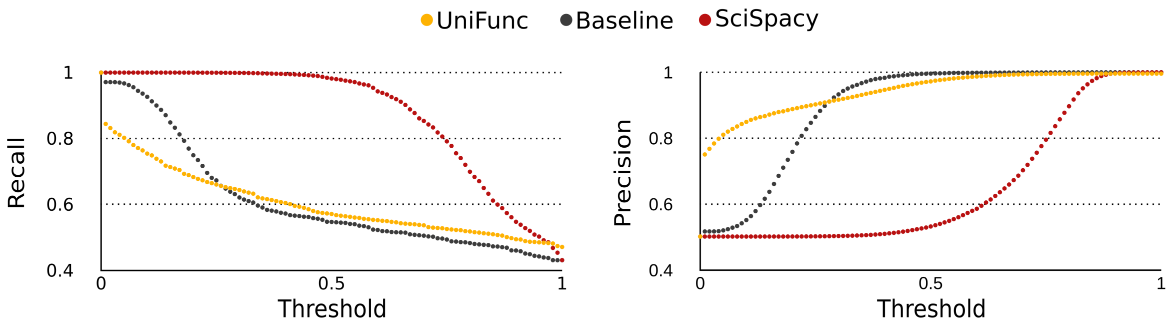


Figure 11 Performance of the baseline and SciSpacy models and UniFunc according to precision and recall

Case studies In the following section, we will provide two case studies that will serve as examples for the potential application of UniFunc. The first entails functionally annotating a sample with multiple reference databases and using UniFunc for integrating these different sources of functional annotations. The second the generation of a profile Hidden Markov Model (PHMM) reference database, using UniFunc to evaluate the functional homogeneity of the PHMM. Methodology details are available in the "**Case studies methodology**" section.

Using multiple reference data sources As an example, we annotated the proteome from the organism *Bacillus subtilis* (Uniprot proteome ID UP000001570) using HMMER [183] against two reference databases (KOfam [6] and NCBI's protein family models [131]).

Using as reference the KOfam PHMMs and NCBI's protein family models (NPFM) we annotated 3324 and 2895 out of 4260 sequences, respectively. Combined, both references annotated 3444 sequences. For each sequence, we checked which functional annotations from KOfam and NPFM shared IDs, those that shared IDs were identified as being functionally equal, of which 492 sequences were found. The remaining sequences would then need to be evaluated according to their functional annotation description; doing so manually would only be feasible for a select number of sequences (not feasible in a high-throughput pipeline). As such, UniFunc was used to evaluate the similarity of functional descriptions and thus aid in the integration of these two reference data sources. Using UniFunc, we calculated the similarity between the functional annotation descriptions from KOfam and NPFM. By setting a similarity threshold of 0.9 (merely an example), we found that 266 were in functional agreement. In this manner, we were able to integrate two reference data sources into a single, non-redundant annotation. A similar protocol could be applied for the annotation of metagenome-assembled genomes or full metagenomes.

Functional homogeneity of a protein cluster During the creation of multiple sequence alignment-based reference data sources, e.g., PHMMs, it is common to cluster protein sequences by their sequence and functional similarity. UniFunc can automate functional similarity analysis. As an example, we evaluated the functional homogeneity and average pairwise sequence distance of a collection (N=4100) of clustered *Archaea* protein sequences. We then kept only the clusters with at least 10 proteins sequences and with a minimum 80% of functionally annotated protein sequences, resulting in 2516 clusters. We then used UniFunc to measure the functional pairwise similarity (i.e., cluster functional homogeneity) between each pair of protein sequences within the clusters. These clusters were further refined by setting a minimum of 0.9 functional homogeneity and a maximum average pairwise sequence distance of 0.1, thus obtaining 2182 highly homogeneous clusters (function and sequence-wise). These new clusters could then be used to create highly-specific PHMMs, which could then be used to functionally annotate *Archaea* samples.

6.2.5 Discussion

We have developed a method that processes and encodes free-text functional descriptions, allowing for high-throughput pairwise similarity analysis, ultimately enabling the comparison of functional annotations obtained from multiple sources. We designed UniFunc with two applications in mind, first to facilitate redundancy elimination when consolidating protein function annotations, secondly as a cross-linking mechanism for functional annotations devoid of IDs. While more sophisticated methodologies have been applied in related fields [126, 232, 154], we aimed to create a method with low complexity that could be easily integrated into more complex annotation pipelines.

UniFunc was developed to identify and eliminate confounders (e.g., determiner ”the”) from functional annotations (i.e., noise reduction). Without noise reduction, annotations such as ”this is an oxidase” and ”this is a kinase” may be indicated as being similar (three out of five identical tokens). On the other hand, without noise reduction, annotations such as ”believed to be an oxidase” and ”this is an oxidase” may be indicated as being dissimilar (two out of seven identical tokens). Fundamentally, confounder elimination increases ”purity” of annotations, resulting in more polarized (close to 0 or 1) similarity scores.

As seen in Figure 11 UniFunc can achieve high precision, in fact, initial convergence to high precision was much faster for UniFunc than for the other models. This is explained by the extreme distribution of the similarity scores, which registers high density toward 0 or 1, whilst in the other models they are more evenly distributed. This extreme distribution of the scores confirms UniFunc’s superior noise reduction ability, which is especially useful when comparing functional descriptions with a high amount of confounders (e.g., when comparing functional annotations with multiple sentences).

Neither UniFunc nor the baseline model achieved high recall at higher similarity thresholds.

This can be explained by the fact that while two functional annotations may imply the same function (and thus share IDs), they may also use different nomenclature (e.g., quercetin can also be called meletin or xanthaurine). In these scenarios, a comprehensive biological lexicon would be required, and while such lexicons exist [222], they are usually behind pay-walls, which limits their use by open source projects following FAIR principles set by Wilkinson et al. [236]. Since several databases (e.g., [30, 36]) provide ontology systems, a future solution may be to use these to create a lexicon from these. Improving UniFunc's NLP (e.g., lemmatization - "methyltransferase" to "methyl transfer enzyme"), could aid in the reduction of false negatives. Among various text mining techniques, word embedding [169, 144] could prove beneficial; this technique permits identifying tokens as similar when they appear in similar contexts. This could prove advantageous in UniFunc's workflow (i.e., specifically during part-of-speech tagging), where instead of removing certain lexical tags (e.g., pronouns), we could use word embedding to understand how related two tokens are (e.g., "enzyme" is closer to "protein" than to "gene"). Word embedding, or similar techniques, could potentially improve UniFunc's recall. However, this added complexity could also reduce UniFunc's robustness when dealing with tokens absent or underrepresented in the annotation corpus (e.g., some tokens appear only once in the whole corpus). In addition, higher complexity techniques also tend to be more costly, both in term of time and hardware requirements.
At higher thresholds, UniFunc had a higher recall than the baseline model, while having similar precision. This is again due to noise reduction and its resulting extreme distribution of scores. Since the baseline model does not eliminate confounders, its ability to produce high similarity scores is reduced, leading to a lower recall when the similarity threshold is too high. SciSpacy and UniFunc behaved very differently along the different thresholds. UniFunc quickly achieved high precision at the expense of a rapid decrease in recall, whereas SciSpacy only achieved a higher precision at higher thresholds. Despite this, the AUC difference between the two model was low, UniFunc's AUC was 0.008 lower than SciSpacy's, in essence, both have their own advantages. SciSpacy is applicable on a broader range of scenarios, whereas UniFunc was specifically implemented to compare protein function annotations. On the other hand, we believe that UniFunc's implementation may offer more

future-proofing as its corpus can be updated by re-downloading Swiss-Prot's protein annotations and gene ontologies.

Overall, all models had an AUC above 0.8, indicating that these performed well in predicting the correct classes of the validation dataset. At a baseline, we consider UniFunc's performance to be consistent (high precision) to allow for its reliable use in the scenarios it was built for. Future iterations will address UniFunc's lower recall.

We have also provided two case studies where UniFunc could be easily applied. In the first, we have shown that UniFunc can aid the integration of multiple reference data sources. This methodology could also prove advantageous when functionally annotating samples with very specific reference data sources (e.g., Resfams [68]). In addition, we have shown that UniFunc can be used to measure functional homogeneity when creating functional reference data sources.

In conclusion, while the standardization of functional annotation data (including IDs and free-text) still constitutes a challenge for successful data integration [92, 121, 205], we have shown that functional descriptions from multiple references/sources can be successfully integrated using text mining, complementing ID-based data integration.

6.2.6 Materials and methods

Baseline model workflow In the baseline model, descriptions are split into tokens by using spaces as delimiters. Each annotation is then encoded according to the presence (1) or absence (0) of all tokens within both annotations (i.e., one-hot encoding). For example, if the first annotation contains the tokens [”lipids”, ”degradation”] and the second [”glucose”, ”degradation”], the union of tokens would correspond to [”lipids”, ”glucose”, ”degradation”]. We then check the presence/absence of the first token ”lipids” in the first annotation: since it is present, we add a ”1” to the first annotation vector. We do the same for the second annotation: since it is absent, we add ”0” to the second annotation vector. We repeat the same process for the second token ”glucose”, so the first annotation vector now is [1,0] and the second [0,1]. We then do the same for all the remaining tokens (i.e., ”degradation”) and obtain, for each annotation, a vector with the number of token as entries. Here, where the first annotation is encoded as [1,0,1] and the second as [0,1,1]. One minus the cosine distance of these vectors will correspond to the similarity of these two annotations.

UniFunc workflow UniFunc's workflow is comprised of four main steps: (i) pre-processing, (ii) part-of-speech tagging, (iii) token encoding and scoring, and (iv) similarity analysis. NLP includes steps i and ii, which were tailored towards the removal of confounder tokens, thus preserving only the most significant tokens within the functional description. Steps iii and iv refer to the text mining part of the workflow and involves the encoding of annotations, making them comparable. Figure 10 shows an overview of UniFunc workflow.

(i) Pre-processing NLP starts with ID extraction via the use of regular expressions in two manners, the first looks for common database IDs (cID) patterns (i.e., enzyme ECs - Cornish-Bowden [42]; TCDB - Saier, Tran, and Barabote [187], KO - Kanehisa and Goto [104]; TIGRfam - Haft et al. [76]; Pfam - El-Gebali et al. [67]; COG - Tatusov et al. [219]; and GO - Consortium [36]), the second finds possible IDs (pID) with the regular expression: ”`[A-Z]+\d{3,}(\.\d+)?([A-Z]+)?`” , which captures ”words” with capital letters followed by a set of numbers (possibly followed by a dot and digits or more capital letters). While ID structure varies, in our experience, this is the most standardized format across multiple databases. Protein acronyms are also captured by this regular expression, which, due to the increased weight of pIDs in comparison to tokens, will increase the similarity score of annotations containing the same protein acronyms. IDs are put aside for later use during similarity analysis.

After ID extraction, the annotation is pre-processed, where unnecessary punctuation and filler entities (e.g., extra spaces) are removed, followed by standardization of nomenclature (e.g., ”`-->`” to ”to”) and numerals (e.g., ”III” to ”3”). Annotations are then split into sentences (sentence segmentation) and each sentence is split into tokens (tokenization, e.g., ”lipid degradation” is split into two tokens ”lipid” and ”degradation”). Each plural token is then converted to its singular form (stemming). Finally, we also aggregate certain tokens (e.g., tokens ”terminal” and ”N” to token ”terminal N”) into a single token. Tokens within parentheses are removed (except when they are acronyms) as, in our experience, they tend to contain tokens irrelevant for similarity analysis (e.g., ”Seems to play a role in the dimerization of PSII (By similarity).”).

(ii) Part-of-speech tagging Part-of-speech tagging (PoST) is the method of lexically classifying/tagging tokens based on their definition and context in the sentence. The aim here is the identification and elimination of tokens that could introduce noise during similarity analysis (e.g., the very common determiner ”the”). We use two taggers, a custom tagger, and NLTK’s pre-trained Perceptron tagger [16, 89]. The first tagger is independent of context and uses Wordnet’s lexicon [145] to identify the most common lexical category of any given token. Should a token be present in Wordnet’s lexicon, a list of potential lexical categories for the token is given (e.g., noun, synonym, verb, etc), the token is then assigned the most common tag. To adjust this tagger’s lexicon to biological data, gene ontologies [9, 36] names, synonyms, and definitions are processed and tagged (using the pre-trained Perceptron tagger). Those not classified as adpositions, conjunctions, determinants, pronouns, or particles are added to the custom tagger as nouns. Tokens are then classified by both taggers, tokens untagged by the custom Wordnet tagger are assigned the Perceptron’s tag. The Perceptron tagger is only used as a backup since it has been pre-trained with the Penn Treebank dataset [220], thus its corpus is unspecific to UniFunc’s target data. Finally, tokens that have not been tagged as adjectives, adverbs, nouns, or verbs, that belong to a pre-compiled list of common biological terms, or are common English stop words are removed from the annotation. Ideally, more tag types would be removed (e.g., adverbs), however, we found that doing so eliminated important tokens. In addition, since some annotations may contain synonym tokens (e.g., a annotation contains the token ”identical” and another annotation contains the token ”equal”), we use Wordnet to find synonyms and replace the respective tokens, such that both annotations contain the same token. This is only applicable to the tokens kept after PoST.

(iii) Token encoding and scoring Token encoding is similar to the baseline's model token encoding, with a key difference, unlike the baseline model's binary vector, UniFunc uses a Term Frequency-Inverse Document Frequency (TF-IDF) scaled vector. Therefore, the annotation vectors will contain only elements with values ranging from 0 to 1, the higher the value the more important the token is in the annotation. TF-IDF measures the importance of each token relative to a annotation and the annotation corpus, therefore tokens that are frequent in a single annotation but infrequent in the annotation corpus receive a higher weight.TF-IDF has been successfully used in past projects, such as Benabderrahmane et al. [13] and Huang, Gan, and Jiang [93].
As a reference corpus, we downloaded all of Swiss-Prot's protein entries (N=563973, as of 2021/01/16) and their respective functional description (”Function [CC]”) and protein names, as well as the gene ontologies ”go.obo” file. From the go.obo file we extracted the ”name:”, ”synonym:”, and ”def:” entries, from the Swiss-Prot file, all the functional descriptions and protein names. This data was then pre-processed (using the same method used by UniFunc) and split into tokens, we then created a frequency table with the number of times each token appeared in the corpus.
TF-IDF is calculated with the equation $\frac{NW}{TW} \times \frac{TC}{NC}$, where *NW* is the number of times a token appears in the annotation, *TW* the total number of tokens in the annotation, *TC* the total number of annotation in the annotation corpus, and *NC* the total number of times a certain token appears in the corpus. We apply a log10 scale to reduce the distance between the vectors' elements and a MinMax scale to sort the tokens by their intra-annotation importance.

(iv) Similarity analysis Each functional description now has an associated set of IDs and an annotation vector. When two functional descriptions share a cID (i.e., enzyme ECs, TCDB, KO, TIGRfam, Pfam, COG, and GO), the similarity score corresponds to 1. When both functional descriptions have pIDs we calculate the Jaccard distance between these sets of pIDs and subtract it to 1, obtaining pIDs similarity pID_{sim}. We then calculate the cosine distance between both annotations and subtract it to 1 to obtain the annotation similarity

Doc_{sim}. If pID_{sim} is above 0, then the similarity score corresponds to $\frac{2 \times pID_{sim} + Doc_{sim}}{3}$, otherwise, it corresponds to Doc_{sim}.

SciSpacy models workflow As an additional comparison we used SciSpacy's [154] "en_core_sci_lg" model. SciSpacy offers biomedical NLP models, which are integrated into the Spacy [90] framework, a NLP Python library. In this model, descriptions are split into tokens by using spaces as delimiters (similarly to the baseline model).

Case studies methodology For the "**Using multiple reference data sources**" case study the functional annotations were generated by using HMMER's *hmmsearch* command against the KOfam and NPFM PHMMs. Since NPFM provides taxon-specific PHMMs, protein sequences were annotated hierarchically, meaning that, if available, we used the PHMMs respective to each taxon of the *Bacillus subtilis* taxonomic lineage (i.e., 131567 $>$ 2 $>$ 1783272 $>$ 1239 $>$ 91061 $>$ 1385 $>$ 186817 $>$ 1386 $>$ 653685 $>$ 1423). In each *hmmsearch* iteration, only the protein sequences left to annotate were used. The functional annotations metadata was then assigned to the hits from HMMER's output.

For the "**Functional homogeneity of a protein cluster**" case study, a collection of 4570 archaeal genomes was downloaded from different sources, 1162 from the UHGG collection of MGnify [146], 371 from NCBI [41], and 3037 from the GEM catalogue [158]. CheckM [162] was run on the entire collection to ensure high-quality archaeomes ($>$ 50% completeness, $<$ 5% contamination). Sourmash [25] was used to identify groups of highly similar genomes in the collection. Similarity of genomes in each group was validated based on their GC content and related taxonomy. Within each group of highly similar genomes, a genome with the highest completeness and lowest contamination was selected as a group representative. As a result, a collection of 1681 non-redundant high-quality archaeal genomes was composed and used to create archaea-specific PHMMs. Protein-coding genes were predicted with Prodigal [98], and functionally annotated with Mantis [176]. MMseqs2 [206] was used to cluster proteins by sequence similarity. Average pairwise distance of each cluster was calculated with Clustal omega [199]. Protein clusters were used to build multiple sequence alignments (MSAs) using MUSCLE [48], and the MSAs were used to construct PHMMs using HMMER [183].

Model validation In order to understand the performance impact of the NLP (context-specific pre-processing and PoST) and encoding (TF-IDF scaled document encoding) approach used by UniFunc, we compared its performance against the previously described baseline model.

As a validation dataset, we started by downloading all the Swiss-Prot [37] entries (N=563973, as of 2021/01/16) with the columns ”Entry”, ”Function [CC]”, ”EC number”, ”Cross-reference (Pfam)”, and ”Cross-reference (eggNOG)”. All the entries with a functional description (”Function [CC]”), and at least one EC number, Pfam ID, and eggNOG ID, were selected, resulting in a validation dataset with a total of 133450 entries.

This dataset allows for the benchmark of each model’s ability to correctly identify similar and non-similar functional descriptions, where similar functional descriptions should have intersecting identifiers (positive class), and non-similar descriptions non-intersecting identifiers (negative class).

Each entry (with a set of IDs S1 and a description D1) in the validation dataset is paired with up to 500 other entries (with a set of IDs S2 and description D2) where $S1 \cap S2 \neq \emptyset$ (positive cases). We then randomly selected an equal number of pairs where $S1 \cap S2 = \emptyset$ (negative cases). As an example, a positive case for the entry ”glucose degradation K01” (S1={K01}) would be ”enzyme that uses glucose as a substrate K01 KO2” (S2={K01,K02}), whereas a negative case for the same entry would be ”lipid degradation KO3” (S2={K03}). In the positive case the ID KO1 is common to both entries ($\{KO1\} \cap \{KO1, KO2\} \neq \emptyset$), whereas in the negative case no IDs are shared ($\{KO1\} \cap \{KO3\} = \emptyset$). Assigning an equal number of positive and negative cases to each entry ensures class balance which validates AUC as a global performance metric [100].

We then use UniFunc and the other models to calculate the similarity score (SS) between each pair of entries’ description. Finally, confusion matrices are created with threshold (T) $\in [0, 1, 0.1]$, where:

- true positives (TP) = $S1 \cap S2 \neq \emptyset \wedge SS \geqslant T$
- true negatives (TN) = $S1 \cap S2 = \emptyset \wedge SS < T$
- false positives (FP) = $S1 \cap S2 = \emptyset \wedge SS \geqslant T$
- false negatives (FN) = $S1 \cap S2 \neq \emptyset \wedge SS < T$

The test case entries are the same for all models. During benchmark, UniFunc does not use IDs for similarity analysis.

6.2.7 Acknowledgements

Author contributions according to the contributor roles taxonomy CRediT was as follows: Conceptualization: P.Q.; Data curation: P.Q.; Formal Analysis: P.Q.; Funding acquisition: P.W. and P.M.; Investigation: P.Q.; Methodology: P.Q.; Project administration: P.Q. and P.M.; Resources: P.Q. and P.N.; Software: P.Q.; Supervision: P.M. and P.W.; Validation: P.Q.; Visualization: P.Q.; Writing – original draft: P.Q. (lead), and P.M.; Writing – review & editing: P.Q., P.N, P.M., and P.W.. All authors proof-read and approved of the content in this research paper. The authors declare that they have no competing interests.
The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [228]. This study was supported by the Luxembourg National Research Fund PRIDE17/11823097.

6.2.8 Concluding remarks

The published paper can be found at `https://doi.org/10.1515/hsz-2021-0125`
The GitHub repository for this tool is available at `https://github.com/PedroMTQ/unifunc`
In this manuscript, the UniFunc tool was described and how it can be integrated into larger workflows, such as Mantis. UniFunc's methodology was described, alongside on how it performs against other methodologies.

6.3 UniFuncNet: a flexible network annotation framework

6.3.1 Summary

The downstream integration of functional annotations often requires techniques compatible with large-scale analysis. A common approach is to use network-based methodologies, which enable the study of biological processes and how they are interlinked within and between organisms. UniFuncNet is a network annotation tool that aims to provide a flexible and hands-free framework for the generation of annotated networks in multiple scenarios. In that sense, UniFuncNet generates networks consisting of four different entity types (i.e., genes, proteins, reactions, and compounds) and many different network generation protocols. These network generation protocols mirror the intrinsic structure of the biological databases (which is also a reflection of biological processes) mined by UniFuncNet, i.e., the structure corresponds to the underlying idea of genes being connected to proteins (through transcription and subsequent translation), how some proteins are connected to reactions, and how reactions involve two or more compounds; to summarise, the structure is the following: gene→protein→reaction→compound. UniFuncNet collects extensive data on multiple entity types and connects these entities in a graph-based manner, thus being more easily used in the downstream analysis.

The authors for this publication[174] are as follows: Pedro Queirós, Oskar Hickl, Susana Martínez Arbas, Paul Wilmes and Patrick May. All authors contributed to the writing, revision, and study design. I was the main author of this publication and the developer of the tool associated with it.

UniFuncNet: a flexible network annotation framework

Abstract

Summary: Functional annotation is an integral part in the analysis of organisms, as well as of multi-species communities. A common way to integrate such information is using biological networks. However, current data integration network tools are heavily dependent on a single source of information, which might strongly limit the amount of relevant data contained within the network. Here we present UniFuncNet, a network annotation framework that dynamically integrates data from multiple biological databases, thereby enabling data collection from various sources based on user preference. This results in a flexible and comprehensive data retrieval framework for network based analyses of omics data. Importantly, UniFuncNet's data integration methodology allows for the output of a non-redundant composite network and associated metadata. In addition, a workflow exporting UniFuncNet's output to the graph database management system Neo4j was implemented, which allows for efficient querying and analysis.

Availability: Source code is available at https://github.com/PedroMTQ/UniFuncNet.

Introduction

There exists an unprecedented amount of biomolecular data available thanks to the advances in, among others, sequencing, mass spectrometry and bioinformatics techniques. This allows for the study of function across several biological levels at high resolution, from single organisms to the combined functional potential of microbial communities. This wealth of information is difficult to access and use in a straightforward and scalable manner, e.g., due to the lack of a universal data repository and the use of a multitude of data formats and annotations.

Networks are frequently used for large-scale omics data analyses as these are versatile tools that can be used to model complex biological systems [115]. The identification and mapping of functional entities (e.g., proteins) to networks are central tasks performed during large-scale studies of new species or microbial communities. For example, networks have been used to study ecological interactions such as metabolic cross-feeding, synergism, and antagonism [152], to detect correlations in metabolic networks [209], to identify keystone functions and genes [185].

Given the available functional annotations linked to omics data, a common modelling approach, among others [64], is to use genome-scale metabolic models (GSMM) to integrate all, or part, of the metabolic and transport reaction network(s) within an organism or community [73, 57]. Such networks are usually derived by mapping functional annotations to the corresponding reactions and pathways [221, 157], and can be used for the *in silico* simulation of metabolism.
Several methodologies [29] and tools [143] are now available for the automated generation and semi-curation of GSMMs. Many methods are able to automatically and accurately reconstruct well-known parts of metabolism, which, due being shared by many taxa [189], have been more extensively studied [114]. While the apparent conservation in function based on homology is advantageous when modelling well-studied metabolism, the resulting GSMMs are often very general and redundant, which may not capture the peculiarities of individual organisms. Modelling species-specific metabolic pathways is important, e.g., for understanding microbial interactions [192], but challenging, since annotations are often incomplete [35, 173]. Here, knowledge integration from multiple databases (e.g., MIBiG [107], KEGG [105], and MetaCyc [31]) may help.

Even though some resources provide frameworks for mapping functional entities (e.g., KEGG [105] and MetaCyc [31]), combining them into a more comprehensive resource at a case-by-case basis is laborious, since this integration requires extensive cross-linking, and often manual review/curation. One additional complication is the use of different ontologies

[210], which leads to the necessity of cross-linking ontology systems with varying structures and resolutions (e.g., KEGG orthologs and gene ontologies[9, 36]). Additionally, while some databases are structured and provide access through the use of application programming interfaces (API) (e.g., KEGG), relational or non-relational or other standardized formats (e.g., json and xml), others provide data in semi-unstructured formats, thereby requiring the implementation of more specialized data processing methodologies (e.g., text mining [218]).

In essence, the diversity and quantity of biological databases, constitute some of the major challenges in the integration of such data. These, and other technical challenges, make such resources inaccessible to researchers without a computational background.
The challenge of integrating knowledge from multiple sources in an automated manner in the context of network analysis was tackled through the development of the presented network annotation framework - (Uni)fied (Func)tional (Net)work (UniFuncNet). UniFuncNet automates the highly time-consuming process of searching multiple databases, extracting and integrating data into a composite output. Biological databases commonly contain multiple entry types (e.g., compounds or genes), therefore, UniFuncNet's implementation reflects the general structure of such databases; for this purpose, we modelled four different entity types: genes, reactions, proteins, and compounds. In turn, these data models can then be linked as a network, and used for storing and exporting information in machine and human-readable formats. Combining data models with multiple data collection methodologies results in a flexible yet robust data retrieval framework. In turn, this allows researchers to fine-tune UniFuncNet to their specific routine data integration tasks, starting from simple use cases such as collecting ChEBI identifiers (IDs) for a list of compound names and finding reactions for certain protein IDs to linking compounds to organisms, or expanding GSMMs. To showcase how the user can include UniFuncNet in their analysis, the last two previously mentioned use cases have been implemented as separate example workflows; while these are simple wrappers around UniFuncNet and other tools, they may serve as a template for future, and potentially more complex, workflows.

UniFuncNet aims to provide a straightforward, versatile, and accessible data collection and network annotation framework. UniFuncNet will prove useful across multiple domains of bioinformatics, especially at a moment in time where large-scale data integration is seen as fundamental rather than optional. In order to provide an easily and efficiently queryable database, we implemented an API that exports UniFuncNet's data to Neo4j.

Materials and methods

Implementation UniFuncNet was implemented in Python (v3.9) and currently collects data from KEGG [105], MetaCyc [31], Rhea [11], ChEBI [80], HMDB [237], UniProt [38] and PubChem [109], cross-linking the information between these databases. For web data collection, UniFuncNet uses the Python package ”requests” (v2.25.1), which queries each database and collects the respective response (usually HTML or json). To parse the HTML responses the ”beautiful soup” [181] package is used (v4.10.0). For some of the databases, i.e., MetaCyc [31], Rhea [11] and ChEBI [80] the database flat files are first downloaded, parsed and stored locally in a SQLite (v3.36.0) database. In order to use the MetaCyc database, the user must obtain a license (academic licenses are freely available) from MetaCyc (which we recommend since it's a highly curated and comprehensive resource). For web data collection, UniFuncNet makes use of API calls to retrieve information (if possible). However, whenever necessary, data is collected by querying the database's website and parsing the query result (i.e., web scraping). Each query result (web or local data) is parsed according to the step of the workflow and database being queried (with database-specific scrapers), and standardized according to UniFuncNet's framework. This data parsing allows for the retrieval of annotations (IDs and synonyms) as well as any connections between database entries.

To avoid overloading the respective web servers, UniFuncNet works in a strictly sequential manner and additionally enforces a time-out for requests to the same server (10 seconds in-between queries by default). Additionally, in order to avoid repeating web queries, UniFuncNet saves past web queries in memory and retrieves the necessary entity whenever a query is repeated. This sequential methodology has the additional benefit of not creating redundant entities which may lead to downstream issues with redundancy and output network connectivity.

The results shown in this paper were collected from multiple sources, MetaCyc version 25.1 was used; the Rhea and ChEBI data corresponded to the flat files uploaded on the 17th of November, 2021; and all data extracted from the multiple websites was collected on the 24th of January, 2022. The version of UniFuncNet used in this paper is v1.02.

Input and output UniFuncNet takes as input a tab-separated file, containing a list of IDs (e.g., ”P02769”), ID types (source of the IDs, e.g., ”uniprot”), entity types (e.g., ”protein”), and search modes (e.g., ”pg”, for ”protein-to-gene”).

UniFuncNet outputs one tab-separated file per entity type, i.e., genes, proteins, reactions, and compounds, listing all the searched entities along with any associated metadata (e.g., database IDs) and all the associations between each entity. Additionally, it outputs a file in simple interaction format (SIF), which allows for integration into network frameworks, such as Cytoscape [196] or Neo4j.

For a detailed description of input format requirements and all outputs, as well as a usage guide refer to UniFuncNet’s documentation at `https://github.com/PedroMTQ/UniFuncNet`.

Workflows methodology We implemented two example workflows to showcase potential applications of UniFuncNet. The first workflow relates to the expansion of GSMMs using UniFuncNet, and the second to the mapping of compounds to organisms. An example use case is provided for each of these workflows. In these use cases, UniFuncNet collected information from the databases KEGG, MetaCyc, Rhea, and ChEBI. The code used for the generation of results is available at `https://gitlab.lcsb.uni.lu/pedro.queiros/benchmark_unifuncnet`. After installation and download of the required tools and data, the workflows are fully automated (e.g., automatically launching tools and doing the necessary data processing). Mantis [177] v1.3 was run for the functional annotation, using the KOfam [6], Pfam [67] and MetaCyc [31] reference databases (the MetaCyc database was generated with the code in `https://github.com/PedroMTQ/refdb_generator`).

Workflow I - Expansion of GSMMs This workflow (Figure 12.A) receives as input multiple protein fasta files, i.e., proteomes, and outputs an expanded network per sample in SIF format. The proteomes are passed to Mantis [177] while GSMMs are created with CarveMe [132]. The enzyme commission numbers (ECs) and MetaCyc protein IDs absent in the CarveMe GSMMs are exported from the Mantis' functional annotations into a UniFuncNet-formatted input file (using the ”prc” search mode). In this manner UniFuncNet can be used to collect data on the additional ECs and MetaCyc protein IDs and connect them to the original GSMMs. In order to exclude unspecific interactions, edges connecting to common cofactors were removed (this list of cofactors has been manually curated but can be edited and is available at `https://github.com/PedroMTQ/UniFuncNet/tree/main/Resources/cpd_to_ignore.tsv`).

The workflow was implemented with CarveMe [132] v1.5. Additional information is available at `https://github.com/PedroMTQ/UniFuncNet/tree/main/Workflows/GSMM_Expansion`. It is crucial to note that this workflow is merely an example use case, therefore any output files created should be thoroughly curated.

As an example application (henceforth referred to as ”use case I”), we used the following five organisms' SwissProt[38] reference proteomes: UP000001031 [227] for *Akkermansia muciniphila*, UP000025221 [201] for *Bradyrhizobium japonicum*, UP000018291 [142] for *Microthrix parvicella*, UP000002528 [70] for *Pelagibacter ubique*, and UP000000586 [91] for *Streptococcus pneumoniae*.

To evaluate the functional redundancy of the baseline and expanded networks, a presence/absence encoding of each network's ECs was applied, followed by a cosine distance calculation using the NLTK package (v3.5), where equal encoded vectors have a score of 1 and completely different a score of 0. This calculation is henceforth referred to as the ”ECs functional redundancy”.

Workflow II - Omics cross-linking. The second workflow (Figure 12.B) attempts to link compounds to specific organisms by searching for information on compounds and linking them to functionally annotated organisms. The input are multiple species proteomes and information (i.e., IDs) on the compounds of interest. The output is a network connecting each compound to all proteomes, and hence, to all organisms. Additional information is available at `https://github.com/PedroMTQ/UniFuncNet/tree/main/Workflows/Compounds_to_Organisms_Mapping`.

As an example application of this workflow (henceforth referred to as ”use case II”), we applied it to the metabolomics dataset MTBLS497 from Metabolights [81]. In the respective experimental study [123] four organisms, *Escherichia coli*, *Klebsiella pneumoniae*, *Pseudomonas aeruginosa*, and *Staphylococcus aureus*, were cultured, sampled and analysed to link them with 13 compounds of interest. The following proteomes from UniProt were used: *E. coli* UP000001410 [233], *K. pneumoniae* UP000000265 [141], *P. aeruginosa* the proteome UP000002438 [211], and *S. aureus* UP000008816 [69].

Results

UniFuncNet UniFuncNet is a network annotation framework that collects user-defined data from multiple biological databases, e.g., KEGG orthology IDs (Figure 13). The user input determines which information is collected by UniFuncNet. UniFuncNet retrieves data from the respective biological databases and parses it; if applicable, it then branches out and gathers any additional data associated with the originally retrieved data. This is repeated iteratively until all sources of information are exhausted. When retrieving information for compounds, UniFuncNet can retrieve data based on synonyms, and not only IDs, as it may facilitate the integration of data where only synonyms are available. However, the reliability of synonyms-based data retrieval is inferior to IDs due to its ambiguity [150]. UniFuncNet is available as a conda package, and it’s respective documentation is available at `https://github.com/PedroMTQ/UniFuncNet`.

Data models In order to standardize the representation of the multiple types of data within the UniFuncNet framework, we implemented multiple data models, each one representing an entity type, i.e., compounds, reactions, proteins, and genes. In general, entities are associated with IDs from multiple databases and other entity-specific data (e.g., com-

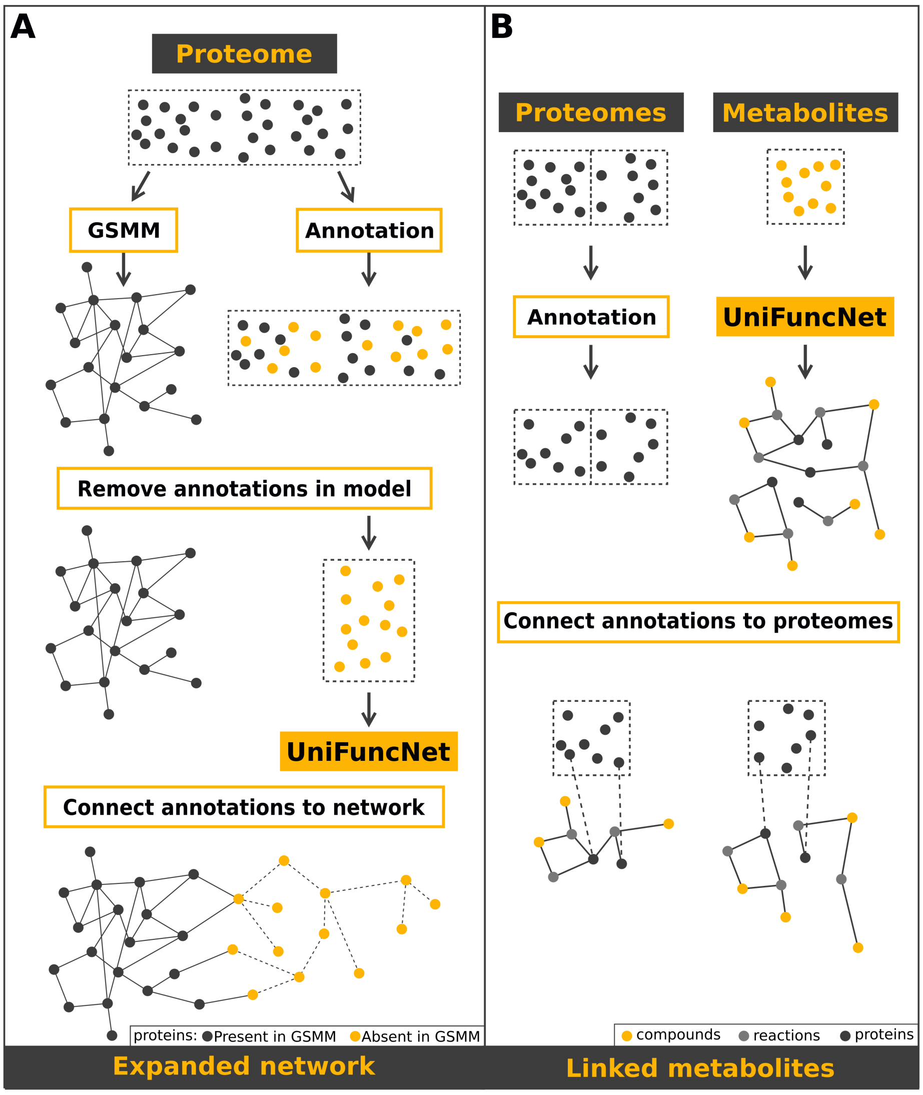
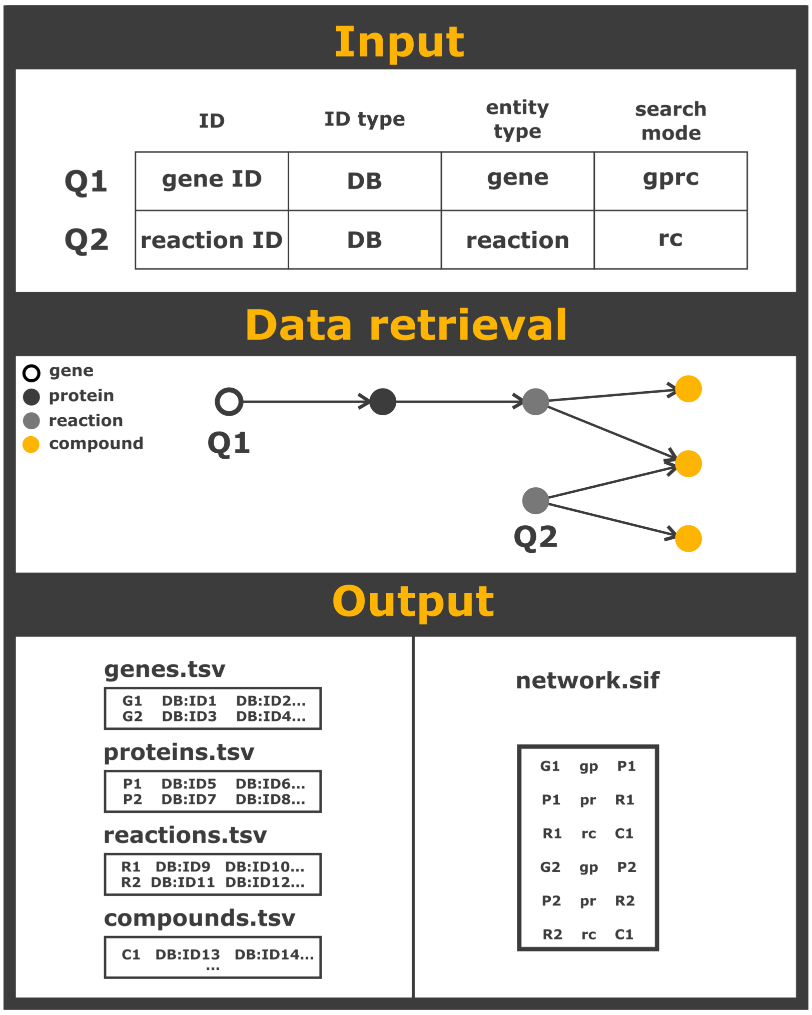

A
Proteome
GSMM
Annotation
Remove annotations in model
UniFuncNet
Connect annotations to network
proteins: ●Present in GSMM ●Absent in GSMM
Expanded network
B
Proteomes
Metabolites
Annotation
UniFuncNet
Connect annotations to proteomes
● compounds ● reactions ● proteins
Linked metabolites

Figure 12 **Use cases workflows. A** Workflow I: UniFuncNet is used to aid in the expansion of a previously generated GSMM. First, a draft GSMM (grey network) and functional annotations (dashed box with grey and yellow nodes) are extracted from the input proteome (top dashed box with grey nodes). Next, all functional annotations absent (dashed box with yellow nodes) in the model are input into UniFuncNet. Lastly, all of the metabolic model's entities are connected to UniFuncNet's output (yellow and grey nodes connected with non-dashed edges). Optionally, the user may also add all remaining nodes in UniFuncNet's network (yellow nodes connected with dashed edges). **B** Workflow II: UniFuncNet is used to identify the proteins of an organism involved in the metabolism of specific compounds. First, proteomes for all organisms were collected (represented by the first dashed box with black dots), these were then functionally annotated with Mantis (represented by the second dashed box with black dots, note the lower number of nodes in each proteome, which represents the lack of functional annotations for some proteins). Using UniFuncNet, we created a network with the reactions and respective proteins associated with each input compound. Lastly, using the previously created network, we linked the compounds with their respective proteins in each proteome.



	ID	ID type	entity type	search mode
Q1	gene ID	DB	gene	gprc
Q2	reaction ID	DB	reaction	rc

genes.tsv

```
G1   DB:ID1   DB:ID2...
G2   DB:ID3   DB:ID4...
```

proteins.tsv

```
P1   DB:ID5   DB:ID6...
P2   DB:ID7   DB:ID8...
```

reactions.tsv

```
R1  DB:ID9   DB:ID10...
R2  DB:ID11  DB:ID12...
```

compounds.tsv

```
C1  DB:ID13  DB:ID14...
       ...
```

network.sif

```
G1  gp  P1
P1  pr  R1
R1  rc  C1
G2  gp  P2
P2  pr  R2
R2  rc  C1
```

Figure 13 **UniFuncNet overview.** The input of UniFuncNet is a list of IDs, ID types, entity types, and search modes, which are processed line by line. In this example, UniFuncNet starts by collecting data on the first query (Q1), which is a gene. According to the search mode ”gprc” it then searches for data for the connected proteins, reactions and compounds. For the second query (Q2) - a reaction, UniFuncNet first collects data on the reaction and then on the associated compounds (search mode ”rc”). UniFuncNet then outputs the results for each collected entity in the respective tsv file, as well as the resulting network in SIF format.

pounds may have an associated chemical formula). The respective data models allow for a standardized in-memory integration, storage, and manipulation of data. For example, the reaction data models are especially helpful for integrating the same reaction from multiple databases; since some reaction database entries do not provide cross-linking, it may be necessary to match reaction entities through their stoichiometry and the compound entities they are associated with (i.e., reactants and products). If the stoichiometry and the reactants and products compound entities are the same, the reactions can be considered the same and merged into the same data model, thus avoiding redundancy. Additionally, these entities can be connected to other entities (e.g., a gene can be connected to a protein), and can thus be exported as a network. Entities are connected within the network following the search mode and databases used (Figure 14).

Search modes UniFuncNet's data models represent the four main entity types ("g" = gene, "p" = protein, "r" = reaction, "c" = compound, see above). These data models are then organized to be retrieved according to the underlying structure of each database; i.e., biological databases entities are generally connected in two directions: $g \rightarrow p \rightarrow r \rightarrow c$ and $c \rightarrow r \rightarrow p \rightarrow g$.

UniFuncNet can process entities in 14 possible search modes, i.e., "gp", "gpr", "gprc", "pg", "pr", "prc", "rpg", "rp", "rc", "cr", "crp", "crpg", "", and "global". Each letter in the search mode corresponds to one of the four different entity types. The "global" search mode corresponds to a search in both directions, e.g., while searching for a given protein, UniFuncNet retrieves information on the associated genes - "pg", as well as the associated reactions and compounds - "prc". The "" search mode corresponds to an "*in situ*" search on the same entity, i.e., UniFuncNet retrieves information on the given input IDs without connecting them to additional other entities, e.g., when one aims to fetch ChEBI IDs from compound synonyms or for ID conversion. Figure 14 represents a generic example of multiple search modes and how these drive network generation.

The user input and search mode are inherently linked to the data that is collected, i.e., the user input ID is used as a seed for data retrieval and to generate an entity, whereas the search mode is used to impose a direction and stop criterion on the data retrieval process. If, for example, the user inputs a reaction ID - the resulting entity will contain the database IDs associated with this reaction. During data retrieval this entity may also be connected to different types of entities, e.g., a reaction entity is usually associated with two or more compound entities. The IDs of these connected entities are then used for posterior data retrieval and generation of the respective entities (Figure 14). The user is able to input multiple search modes (comma separated) for the same input ID, which may be useful, e.g., for connecting a reaction entity to its respective compound and protein entities.

UniFuncNet to Neo4j API In order to provide users with the possibility to efficiently query and manage the UniFuncNet results (for example during network analysis), an API importing UniFuncNet's output into Neo4j a highly-flexible graph database management system, was implemented.

UniFuncNet's output can be depicted as a multipartite graph, which is a graph whose nodes can be split into multiple independent sets. In the case of UniFuncNet each output file contains multiple entities (e.g., proteins) with entity related annotations (e.g., ECs) (Figure 15). Since Neo4j is a highly flexible graph-based database it provides a natural integration of UniFuncNet's data models.

The API takes as input a folder containing all the UniFuncNet output tsv files and stores the data in a Neo4j database. This database can then be queried using Cypher (Neo4j's querying language) or using any programming language Neo4j API (e.g., the Python or Java drivers). Additionally, we added the option to input Mantis consensus annotations to query the Neo4j database and create the respective SIF networks.

Use cases UniFuncNet is a flexible network annotation framework, being usable within diverse contexts. We provide two case scenarios, the first using UniFuncNet for the expansion of GSMMs, and the second for linking compounds with specific organisms.

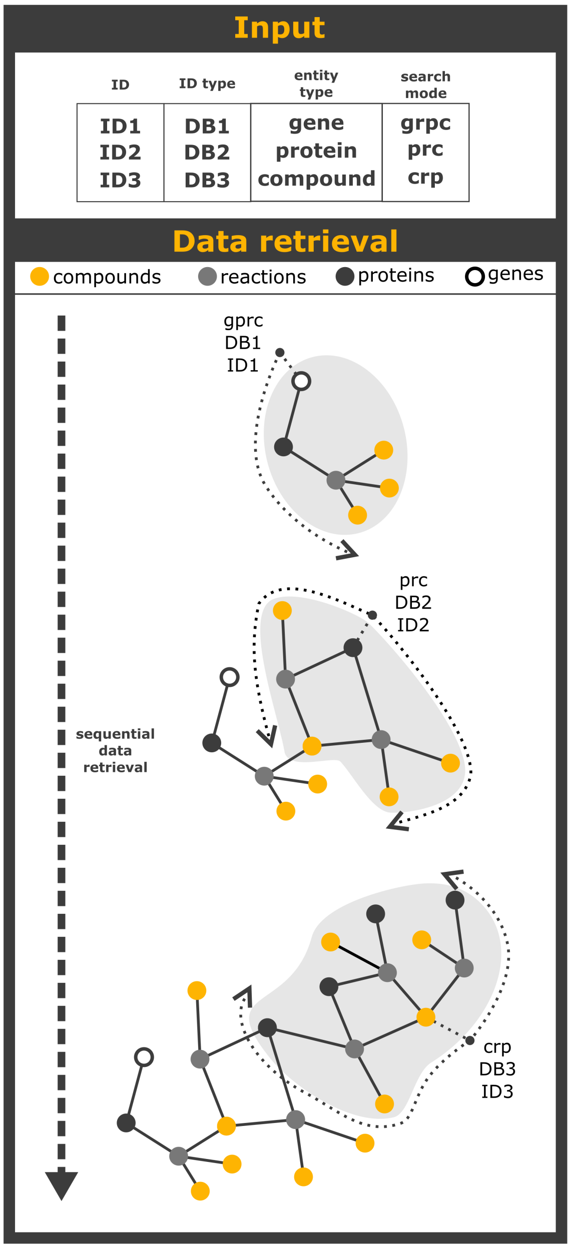


Figure 14 **UniFuncNet search modes**: Example of three different search modes available in UniFuncNet and how they sequentially link entities, generating a connected network. The first input line contains a gene ID with search mode ”gprc”, UniFuncNet searches first for information on this gene and subsequently the directly or indirectly connected entities (one protein, one reaction and three compounds). The second input line contains a protein ID with the search mode ”prc”. UniFuncNet retrieves first information on the protein, then on two reactions and four compounds; notice how one of the compounds found in the second search is linked to the network created already during the processing of the first input. The third input line contains a compound ID, and the search mode ”crp”, UniFuncNet then retrieves information on four compounds, three reactions and four proteins. Again, since one of the proteins was already searched during the processing of the second input line, the resulting network will connect these inputs’ entities.

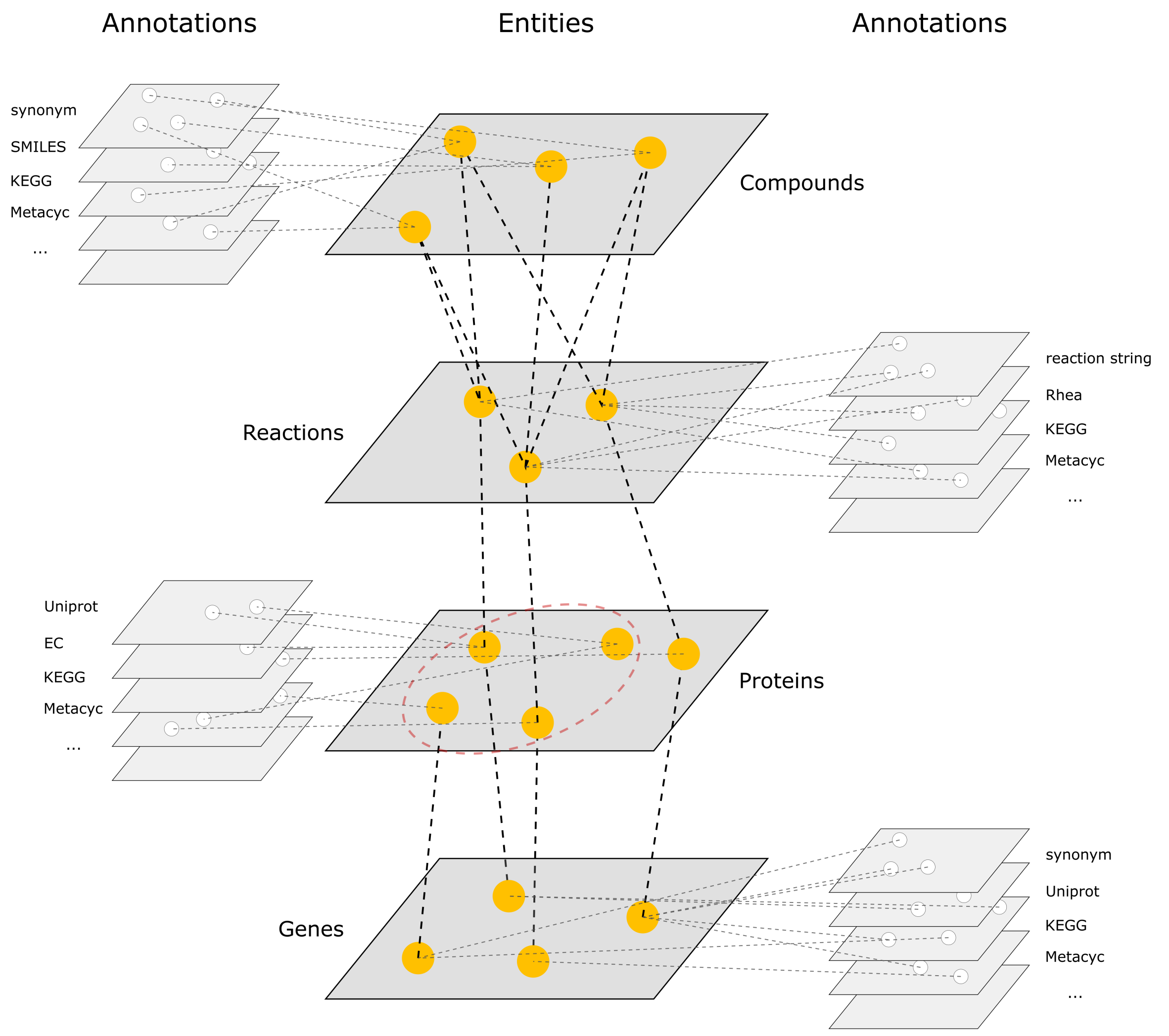


Figure 15 **UniFuncNet results as a multipartite graph.** The output from UniFuncNet can be represented as a multipartite graph, where the central layers correspond to the entity types (e.g., proteins), and the outer layers to the annotations (e.g., IDs or synonyms). The protein layer contains a protein complex (red dashed circle), comprised of multiple subunits (i.e., protein nodes).

Use case I In this use case we used UniFuncNet to expand GSMMs built with CarveMe[132], exploring how many putative connections UniFuncNet could add to the original GSMM. To that end, Mantis is used to provide additional functional annotations, and UniFuncNet to map those functional annotations to the GSMM (Figure 12.A).

As an example, we expanded the GSMMs of multiple organisms that have been shown to be relevant in multiple ecosystems; (i) *Akkermansia muciniphila* has been shown to play an important role in human intestinal health as part of the gut microbiome [160]; (ii) *Bradyrhizobium japonicum*, has been shown to be a key organism in nitrogen fixation, essential for e.g. soybean plant growth [84]; (iii) *Microthrix parvicella*, has been shown to be the dominant species involved in the bulking of activated sludge and lipid accumulation in wastewater treatment plants [197]; (iv) *Pelagibacter ubique*, has been shown to be an ubiquitous ocean-dwelling bacterium that belongs to the SAR11 clade, which is reported to account for 25% of all cells in the ocean [70], and (v) *Streptococcus pneumoniae*, has been shown to be a key human pathogen, which is one of the leading causes of pneumonia, bacterial meningitis, and sepsis [21].

This workflow compiled a list of all ECs and MetaCyc protein IDs found by Mantis that are not part of the original GSMM (generated by CarveMe). A non-redundant list of IDs over all species was generated. This list was converted to a UniFuncNet input file, which contained 1329 unique EC numbers and 1052 unique MetaCyc protein IDs. UniFuncNet were then run for all these IDs with the ”prc” search mode to connect the (p)rotein function annotations to the (r)eactions and (c)ompounds.

UniFuncNet collected 5244 putative reactions, which were then filtered according to multiple steps: (i) filter for proteins associated to at least one reaction (ii) filter for proteins that were also absent in the original GSMM (iii) extract all reactions connected to these proteins, (iv) exclude reactions that were already present in the original GSMM, and (v) match the compounds obtained from UniFuncNet with the GSMM compounds to match reactions and exclude redundant reactions.

For each proteome, a baseline directed network from the initial GSMM was created, where reactions and their respective substrates and products are represented as nodes (i.e., reactant(s)$\rightarrow$reaction$\rightarrow$product(s)). We then expanded the network by adding UniFuncNet's nodes, either by adding new connections to the baseline network or adding new nodes.

The draft GSMMs and expanded networks were evaluated according to: (i) % of reactions in the largest network component (%RLC); (ii) % of dead end metabolites (%DEM), i.e., metabolites without a transporter reaction that are produced but not consumed or consumed but not produced [133]; (iii) % of newly connected dead end metabolites (%CDEM); and (iv) the amount of new putative reactions that could be added to the GSMM.

On average %RLC decreased from 99.6% (sd=0.4%) to 95.0% (sd=0.7%), &DEMs increased from 4.2% (sd=0.6%) to 12.8% (sd=0.6%), and 0.1% (sd=0.06%) of DEMs were successfully connected in the expanded network. Finally, on average 1005 (sd=485.5) reactions could be added per proteome.

The expanded networks resulted in a substantial enzyme-specific enrichment (i.e., ECs), the most enriched ones being transferases, oxidoreductases and hydrolases. We also analysed the ECs functional redundancy of the each organisms' baseline and expanded networks, i.e., each baseline network was compared to all others baseline networks, and the same was repeated for the expanded networks. On average, we found that the ECs functional redundancy for the "only baseline", "only expanded", "baseline+expanded" networks was 0.74, 0.44, and 0.66 (0-1, 1 being equal), respectively.

When analysing KEGG pathways, the most enriched metabolic capacities corresponded to the metabolism of carbohydrates, lipids, and cofactors and vitamins (from least to most enriched). In the *Akkermansia muciniphila* expanded network, the biosynthesis and metabolism of glycans was among the metabolic capacities most enriched by the network expansion (161 ECs in the baseline to 166 additional ECs in the expanded network mapped to the kegg pathway "Glycan biosynthesis and metabolism"). In the *Microthrix parvicella* expanded network, the metabolism of lipids was the metabolic capacity most enriched by the network expansion (31 ECs in the baseline to 218 additional ECs in the expanded network mapped to the kegg pathway "Lipid metabolism").

These results are available in Supplementary table "results.ods" available at `https://gitlab.lcsb.uni.lu/pedro.queiros/benchmark_unifuncnet`.

Use case II In order to understand how UniFuncNet could be used to link different omics levels, we used it to connect functionally annotated reference proteomes to a metabolomics dataset, i.e., linking metabolism related proteins to their reactions and respective compounds (Figure 12.B).

As an example, we used a metabolomics study [123] that cultured four organisms in artificial sputum and nutrient broth mediums and sampled their headspaces for 13 compounds. These compounds were used as potential biomarkers in order to determine the most appropriate antimicrobial therapy in the treatment of ventilator-associated pneumonia.

In order to find the reactions and proteins associated with each compound, UniFuncNet ran with the search mode ”crp”. The proteins found to be connected with the compounds via UniFuncNet were then intersected with the functional annotations of each proteome, thus allowing for the identification of the enzymes within each organism involved in the metabolism of these compounds.

After running this workflow we successfully retrieved information on 11 of 13 compounds, eight of these were linked to a total of 30 reactions. These reactions were then connected to a total of 17 proteins. We then linked the proteins connected to reactions (n=17) to the functional annotations of each organism, finding which of these organisms could potentially be involved in the metabolism of studied compounds. We found that all organisms were involved in the metabolism of indole and that *Pseudomonas aeruginosa* was additionally involved with the metabolism of 2-furanmethanol.

Discussion and conclusion

Here we present UniFuncNet, a network annotation framework that collects and integrates data from multiple biological databases. UniFuncNet can be used to search for information and generate annotated networks in a flexible manner (i.e., various search modes and input ID types). UniFuncNet automates data collection into a human-readable output, by connecting the different biological entities (i.e., genes, proteins, reactions, and compounds), and it provides a network-structured output, which can be easily used in network-based downstream analysis. An added benefit of UniFuncNet is the standardization of the search methodology, potentially decreasing the accidental omission of information during manual collection/curation.

UniFuncNet collects data from live websites/application programming interfaces and allows the user to update their own local flat files (e.g., MetaCyc or Rhea). UniFuncNet ensures that the collected data is up to date, which represents a limitation in similar projects [150]), since they require regular database maintenance. However, UniFuncNet faces its own challenges: (i) a website's HTML structure or API may change over time, which requires maintenance of UniFuncNet's data collection protocols, (ii) live retrieval of information tends to be slower than using a centralized source of data, and (iii) websites may block scraping attempts if these are done too frequently, which is circumvented by UniFuncNet by having 10 second waiting period between each web query to the same database. While reliable and large data collection is provided by UniFuncNet, as a framework that can speed-up the work of researchers requiring comprehensively annotated networks, it is advisable to perform manual curation during downstream data integration. Overall though, we believe that the benefits of having a lightweight framework with very low storage footprint, always retrieving the latest information, clearly outweigh the aforementioned downsides.

Current automated reconstruction tools are capable of generating GSMMs ready for modelling. However, divergent implementations [246, 132, 47] may lead to different outcomes (i.e., the models) due to multiple factors, e.g.: (i) different gene predictions, (ii) different functional annotation reference databases, and (iii) different automated curation implementations [56, 82]. While automated curation offers a good modelling basis, it is unlikely that the current methods will ever be able to encompass the complexity of *in vivo* biological networks. As such, manual curation and expansion of GSMMs remain essential; the latter is routinely done through the iterative analysis of the subsystems for genes, proteins, and reaction(s) of interest. In particular, the end-user searches for information regarding a certain ontology ID, such as KEGG [105] orthology IDs, ECs, or others, in highly comprehensive (and partially redundant) biological databases. To avoid introducing redundancy, cross-linking entities between databases is necessary, which can be done manually or partially automated through ID mapping tools offered by MetaNetX [150] or UniProt [38]. To this end, we implemented a workflow that uses UniFuncNet to facilitate the cross-linking and expansion of GSMMs.

We have shown that the networks enriched with UniFuncNet's workflow were better able to capture organism-specific characteristics, e.g., in *Microthrix parvicella* the metabolism of lipids was the most enriched KEGG pathway, which agrees with the findings of Sheik et al. [197]. Similarly, in *Akkermansia muciniphila* the metabolism and biosynthesis of glycans was amongst the most enriched KEGG pathways, which supports the hypothesis that this organism and glycans play an important role in human gut health [160, 113]. Lastly, the metabolism of cofactors and vitamins was, on average, the most enriched KEGG pathway among all organisms.

In general, we found that this workflow could add a substantial amount of reactions to the GSMMs, which, as previously shown [177], is likely due to the more comprehensive reference databases used (Mantis with the KOfam, Pfam, and MetaCyc databases and CarveMe with the BIGG database [191]). In addition, we also found that the similarity between the functional profiles (i.e., ECs functional redundancy) between each organism's network was substantially lower in the expanded networks, highlighting the benefit of applying UniFuncNet to discover functions unique to each organism. It is important to emphasize that the expanded networks would still require curation; indeed the aim of this workflow is not to directly output an expanded GSMM ready for modelling, but to provide the user with a framework that automates some of the most time-consuming curation steps, i.e., expanding and enriching the network. Altogether, these results show the potential of UniFuncNet to support the expansion of GSMMs, provided additional curation steps are implemented by the end-users. While in this manuscript we have shown how UniFuncNet can be used in a targeted manner, it could also be used for the generation of genome-scale metabolic networks.

We have also shown how UniFuncNet can be used to link different datasets, in particular how it can be used for linking different omics, which should prove useful for multi-omics network-based analysis. Specifically, in the second workflow, we have shown how UniFuncNet may be used for the mapping of compounds to specific organisms. The results shown in the use case II were not able to connect the organisms and compounds in the same resolution as the respective study [123], which further highlights the need to create and use more comprehensive functional annotation reference databases. However, we found that indole's metabolism was shared among all organisms, which is a clear indication of conservation of function in prokaryotes[134, 223, 241]. Despite this, we believe this workflow could be combined with more resolved input proteomes (i.e., using proteomics data instead of reference proteomes) and as such could be an even more powerful screening tool for more thorough investigations.

In conclusion, in this article we have highlighted UniFuncNet's ability to automatically and comprehensively annotate networks. Additionally, we have showcased two use cases which could be used as baseline examples for more intricate analysis. We believe that UniFuncNet's flexible search modes and varied input formats expands its utility into a variety of analysis well beyond the ones shown in this paper.

Acknowledgements

All authors proof-read and approved of the content in this research paper. The authors declare that they have no competing interests. We would like to acknowledge Ines Thiele and Alberto Noronha for their supervision during the initial stages of this project. Supported by the Luxembourg National Research Fund PRIDE17/11823097 and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 863664).

Conflict of interest statement.

None declared.

6.3.2 Concluding remarks

This manuscript has been submitted and the pre-print is available at https://doi.org/10.1101/2022.03.15.484380 The GitHub repository for this tool is available at https://github.com/PedroMTQ/unifuncnet

In this manuscript, UniFuncNet was described and how it can be used to generate highly annotated networks, integrating knowledge from multiple databases into one composite network. Two case studies and respective workflows were showcased, which I believe could be useful within their target community; moreover, they could be used as a basis for additional workflows. Finally, an API that uses UniFuncNet's output to automatically generate a Neo4j database was developed. This database can be easily queried in high throughput analysis.

7 Discussion

In this thesis, I have shown how relevant data integration is within different bioinformatic domains, particularly at the speed and quantities in which data is currently generated. I believe data integration will be increasingly more important in the years to come. In that regard, it is expected that new tools and methodologies for the integration of biological data will be created. With that said, I believe it is important to discuss some of the issues with the current tool development environment in bioinformatics. The harsh ”incentive” to publish (tools) without regards to maintenance and quality has led to the creation of hundreds of one-time use bioinformatics tools [137, 136]. Indeed, this proliferation of tools without quality control exacerbates many of the issues felt by the end-users of those same tools, such as: (i) tool is not open-access, (ii) tool is not properly archived or versioned, (iii) tool cannot be installed or run, (iv) tool is not maintained, and (v) authors do not provide user support. This of course happens for many reasons, e.g., lack of funding, the absence of a plan to maintain the tool, or lack of expertise to implement production-ready software (or a meagre semblance of it). Despite the grim criticism, this has been steadily improving in the past years; e.g., peer-reviewed journals now require higher standards for published software, there is more emphasis on reproducibility, there are more software frameworks supporting researchers (e.g., Snakemake [148] and Nextflow [46]), and more and better training resources are available. Naturally, as the field of bioinformatics matures, so will the standards regarding software production.

7.1 Software improvements and future work

Sustained software development provides major benefits [65] (e.g., accuracy), to contribute to this end (i.e., software maintenance), I have tried to continuously support users (answering users within a reasonable time frame and implementing user requests) and also introduced major features in the software developed. I hope I am able to provide maintenance and user support for years to come.

7.1.1 Mantis

The Mantis tool has seen considerable development since publication, new major features were added and efficiency improvements were made. Specifically, the following was implemented:

1. ability to use Diamond [26] for homology search; while the original version of Mantis included only HMMER as a method for homology search, Diamond is now also supported (sequence-based homology search). This is highly relevant as it allows for the use of reference databases where PHMM are not available; this can be simply due to the fact that the reference database has an insufficient number of sequences to build proper MSAs (and, by extent, PHMMs), or because it has a resolution better suited for sequence homology-based search. The addition of Diamond brings more versatility to Mantis, which I believe will make it a more widely used protein function annotation tool.

2. the NOG reference database implementation was changed so that both PHMMs or Diamond databases can be used. This is an important improvement since the NOG PHMMs had limited use due to their size (around three terabytes). In addition, the NOG PHMM reference database transferred the function from multiple sequences, resulting in an PHMM with more noisy functional annotations (i.e., more FPs).

3. the option to generate a KEGG module completeness matrix for each sample (user request) was added. KEGG is a commonly used tool for pathway-based downstream analysis. To do so, Mantis compiles a list of all the KOs annotated on any given sample and attempts to generate the most likely KEGG module pathway [54] (since multiple KO pathways per module are possible). A KEGG module completeness score is then output (the number of KOs in the sample divided by the total number of KOs for the best KO pathway of any given KEGG module) per sample, which can then be used for comparative analysis.

4. a GTDB [163] to NCBI taxa converter (and vice-versa) was implemented. Previously, Mantis only accepted the input of NCBI IDs (these are used to determine which taxa-

specific reference databases to use), it now accepts both NCBI and GTDB taxa, which should provide more input versatility. Since some taxa from NCBI and GTDB may be ambiguously mapped to multiple taxonomic lineages, an algorithm to determine the last common ancestor for all the possible lineages was implemented.

5. Metadata (i.e., the functional descriptions and IDs associated with each entry in the reference database) is now stored in SQLite databases, which increases the efficiency at which metadata is associated with each hit. This is especially important for databases with metadata files containing multiple gigabytes of data.

6. mantis is now installable through conda at `https://anaconda.org/bioconda/mantis_pfa`

7. addition of the TCDB [188] transporters database and removal of TIGRfams (since these are included in the NCBI protein family models).

While the default reference databases used by Mantis are quite comprehensive, they may not be applicable in more specific scenarios. For this reason, a small repository that automatically creates additional Mantis-compatible reference databases was created. These tend to be more context-specific and have therefore not been added to Mantis. These include Rhea reactions, Reactome reactions, and EC PHMMs, as well as BIGG genes, SwissProt and Trembl diamond databases. To note that each database used in the creation of these references required the implementation of different methodologies for data extraction. PHMMs are created by clustering sequences by the respective ID (e.g., for Rhea PHMMs, we group all sequences for a given Rhea reaction ID into one fasta file), and posterior similarity clustering with mmseqs2 [207] (e.g., splitting the Rhea ID specific fasta file into multiple fasta files, based on how the sequences cluster). This project is available at `https://github.com/PedroMTQ/refdb_generator`.

While Mantis provides significant improvements in the associated field through its highly flexible and consensus-driven protein function annotation, it is also limited by the methodologies that it uses, i.e., homology-based methods. Indeed, there is only so much time and funding available for experimental validation, only so many appropriate experiments for the

generation of reference databases, and only so much that can be done until computationally-generated reference databases become too far-fetched. Despite these limitations, homology-based methods will likely remain the basis for function annotation for several more years.
Despite this, we can not ignore the major improvements in the methods to predict structure, which could be particularly useful for functionally describing sequences unsuccessfully annotated by homology-based methods. In the field of structure prediction, two major tools should be highlighted - AlphaFold [103] and trROsetta [212]; these two tools are at the forefront of structure prediction through the use of deep learning.
Deep learning is a sub-field of machine learning that uses artificial neural networks. Deep learning models generally have a multi-layered structure, i.e., an input layer, hidden layer(s), and an output layer [124]. These models require a large amount of training data (and time) and are generally challenging to train, not only due to their lack of interpretability, but also because they are very hardware dependent (due to the scale of data required). Despite some drawbacks, when large amounts of data are available, deep learning tends to outperform more traditional machine learning [1]. In recent years, the application of deep learning has exploded in biology and other fields [190].
Deep-learning-based structure prediction tools have recently been brought to the spotlight as substantial advances in the field have been achieved by non-academic parties, e.g., AlphaFold [103] by Deepmind, a subsidiary of Alphabet/Google. Methods like AlphaFold could allow for the analysis of biological data not currently annotated by more conventional methods [51].
Since homology-based methods may sometimes fail to functionally annotate sequences (insufficient reference data), it may be interesting to use AlphaFold to predict the structure of these sequences (especially for niche taxa). For example, one could create a workflow that would receive a list of protein sequences; these would then be annotated with Mantis, and those that are not annotated would have their structure predicted with AlphaFold. One could then take this structure prediction and use tools such as ProFunc [122] to associate a function to the structure. ProFunc is a server for predicting protein function from 3D structure via the use of sequence scans (e.g., sequence search in PDB [14]), fold and structural motifs,

and n-residue templates.

7.1.2 UniFunc

The UniFunc tool was developed within the scope of Mantis, i.e., it was developed so that consensus annotations could be extracted from the multiple databases used by Mantis. Due to its scope, this tool has not seen any major improvements nor did it require major maintenance. UniFunc is now also installable through conda at `https://anaconda.org/conda-forge/unifunc`

While it is unlikely that major improvements will be implemented, UniFunc could benefit from a larger lexicon, as well as more sophisticated methodologies for evaluating the similarity of functional descriptions. While UniFunc's lexicon is already quite comprehensive (it contains data from eggNOG, Pfam, KOfam, NCBI, GO, and UniProt and uses the respective metadata as a corpus), additional databases could be potentially added. More importantly, UniFunc would also benefit from a dictionary-based system containing tokens and respective synonyms, which would improve UniFunc's ability to handle nomenclature discrepancies. Such dictionaries are especially hard to create due to the heavily technical nature of functional descriptions. Such highly specific lexicons are scarce or behind paywalls [222], prohibiting their use; manually replicating such work would require considerable expertise and time investment. A potential approach to do this automatically would be to gather a large corpus of functional descriptions and mine them for synonyms based on token associations (e.g., using a graph-based approach [2]). Another potential approach would be to improve the encoding of functional descriptions, in particular, one could use a word embedding model to encode the pre-processed functional descriptions and use this instead of the current approach which doesn't capture contextual information. This would potentially eliminate the need to use a lexicon for synonym retrieval, instead, the word embedding could create similar vectors for tokens that appear in similar contexts. Finally, while UniFunc aims to provide an interpretable and scalable approach to similarity analysis, more complex (and potentially more widely applicable) NLP methodologies could be used . For example, Google's Natural Language API could be used for PoST, instead of the currently used Perceptron tagger.

7.1.3 UniFuncNet

In order to integrate functional annotations (coming from Mantis) into graph-based downstream analysis, a network annotation tool was created - UniFuncNet. UniFuncNet integrates data from multiple databases and structures this data into a network. To showcase how versatile UniFuncNet is, two workflows that use UniFuncNet were also implemented; I believe these will not only be useful to the community but also hopefully serve as a foundation for users to build their workflows.

While UniFuncNet is a flexible tool, it has one major flaw - it does not contain any proper data storage framework (i.e., it exports data to a tsv), which reduces scalability and reusability in more complex downstream analysis. A solution to this problem is to take the data output by UniFuncNet and store it in relational or non-relational databases (e.g., SQL, MongoDB, etc). Since UniFuncNet's data inherently has a network topology, a good solution is to use a graph-based non-relational databases such as Neo4j.

Another major limitation of UniFuncNet is the fact that it requires web scraping (i.e., collection of data from website), which can be inefficient, but, more importantly, requires continued maintenance (since websites continuously change their structure). In that regard, it would be optimal to collect data from downloadable files that include the whole database (i.e., database dump); which could then be parsed and integrated into a composite framework. Unfortunately, some of the databases used by UniFuncNet require a subscription to access these database dumps. In any case, since UniFuncNet's framework is quite flexible (e.g., Rhea and MetaCyc are parsed from their database dumps, while the others are web scraped), additional databases could be added to UniFuncNet, either through web scraping or parsing of downloaded files.

7.2 Application of work

It is important to highlight the applications of the work developed during this PhD, after all, the aim of software development is for it to be used by the respective audience. In that regard, I will now go over some of the applications of the software developed, these include how the community applied this software and also how I applied it within other projects.

7.2.1 Mantis

At the time of writing, the Mantis publication has been cited 8 times, and the respective GitHub project has been starred by 34 people. Mantis is being used in a few different projects:

- it was used for the functional comparison of different omics domains in the publication ”Critical Assessment of Metaproteome Investigation (CAMPI): A Multi-Lab Comparison of Established Workflow” [226]
- it is used by the binning tool ”binny” [86] for functional annotation with a marker gene reference database.
- it is used in the scope of a metabolic modelling tool for the functional annotation of genomes using the BIGG genes diamond reference database (submission pending).
- it was used for the functional annotation of Archaeal proteins, so that, in conjunction with AlphaFold[103], it could be used for better resolution of functions of unknown Archaeal proteins (submission pending).
- it was used to functionally annotate the lipid accumulating organisms coming from wastewater treatment samples
- it is used in the in-house developed multi-omics processing and analysis pipeline IMP [153]

7.2.2 UniFunc

UniFunc was initially created in order to allow for the creation of consensus annotation with Mantis, i.e., similarity analysis of functional descriptions between the multiple databases used by Mantis. Additionally, per a user request, a new workflow that uses UniFunc was created. This workflow analyses the functional annotations of orthogroups (i.e., ”set of genes that are descended from a single gene in the last common ancestor of all the species being considered” [53]) and selects a representative function per orthogroup. This is done in the following manner:

1. parse orthogroups (i.e., cluster IDs), gene IDs and respective gene annotations
2. compare intra-orthogroup functional annotations in a pairwise manner
3. build clusters of functional annotations per orthogroup (identified as similar by UniFunc) and calculate the intra-cluster functional similarity, which should indicates how coherent (in terms of function) the cluster is
4. obtain counts for all functional annotations per orthogroup, which indicates how many times a certain function appears in the orthogroup
5. scale the functional clusters similarity (min-max scaling), so that is within the range 0 to 1
6. sum the functional annotations counts per cluster to obtain the total counts per cluster. Scale the cluster counts (min-max scaling) so that it is also within the range 0 to 1.
7. average the functional cluster similarity and counts to obtain a functional cluster score
8. select the highest functional cluster score as the representative function for the orthogroup

7.2.3 UniFuncNet

As previously mentioned, it would be beneficial to store UniFuncNet’s data within a database. In this regard, an API that receives data from UniFuncNet, and stores it in a Neo4j database (available at https://github.com/PedroMTQ/UniFuncNet/tree/main/Workflows/UniFuncNet_Neo4j_Connector) was implemented. This API was designed to store UniFuncNet's multipartite output graph into a Neo4j database, i.e., four main node types exist (one for each entity type - genes, proteins, reactions, and compounds); similar to how UniFuncNet entities contain information such as database IDs and connections to other entities, this Neo4j database also contains such associations, however, these are now done by generating sub-nodes that are connected with the main node types. Please refer to Figure 15 for a visualisation of this multipartite graph and Figure 16 for an example on how this data is represented in the Neo4j database.

Since UniFuncNet can dynamically generate annotated networks (depending on the user input), this database's structure depends on the output generated by UniFuncNet. In order to create a generic database that integrates data from multiple sources a list of 6614 ECs, 5935 KOs, 13886 Rhea reactions and 20795 MetaCyc protein IDs into a single UniFuncNet input file (available at https://github.com/PedroMTQ/UniFuncNet/tree/main/Workflows/Input_Generator) was automatically compiled. This file was then used by UniFuncNet to generate a total of 27824 protein, 19850 reaction, and 21064 compound entries. This data was then fed into the Neo4j database to create a database with 502992 nodes (entities as main nodes plus entity annotations as sub-nodes) and 1445108 edges.

This database has since been tested with two different end goals in mind: (i) creation of organism-specific functional networks and (ii) binning refinement.

Using this Neo4j database it is possible to create functional annotation derived networks, i.e., functional annotations are extracted (from Mantis' output) and used to query the database. By extracting protein-reaction-compound connections, a functional network is created, which is then exported in SIF format (other formats could be added). This network can then be studied or visualized by different network analysis tools. This workflow is available at https://github.com/PedroMTQ/UniFuncNet/tree/main/Workflows/UniFuncNet_Neo4j_Connector. It would then be interesting to complementarily use different meta-omics layers and implement a network-based downstream analysis; e.g., (i) functionally annotate MAGs with Mantis (ii) generate MAG-specific (or community-wide) functional networks, (iii)

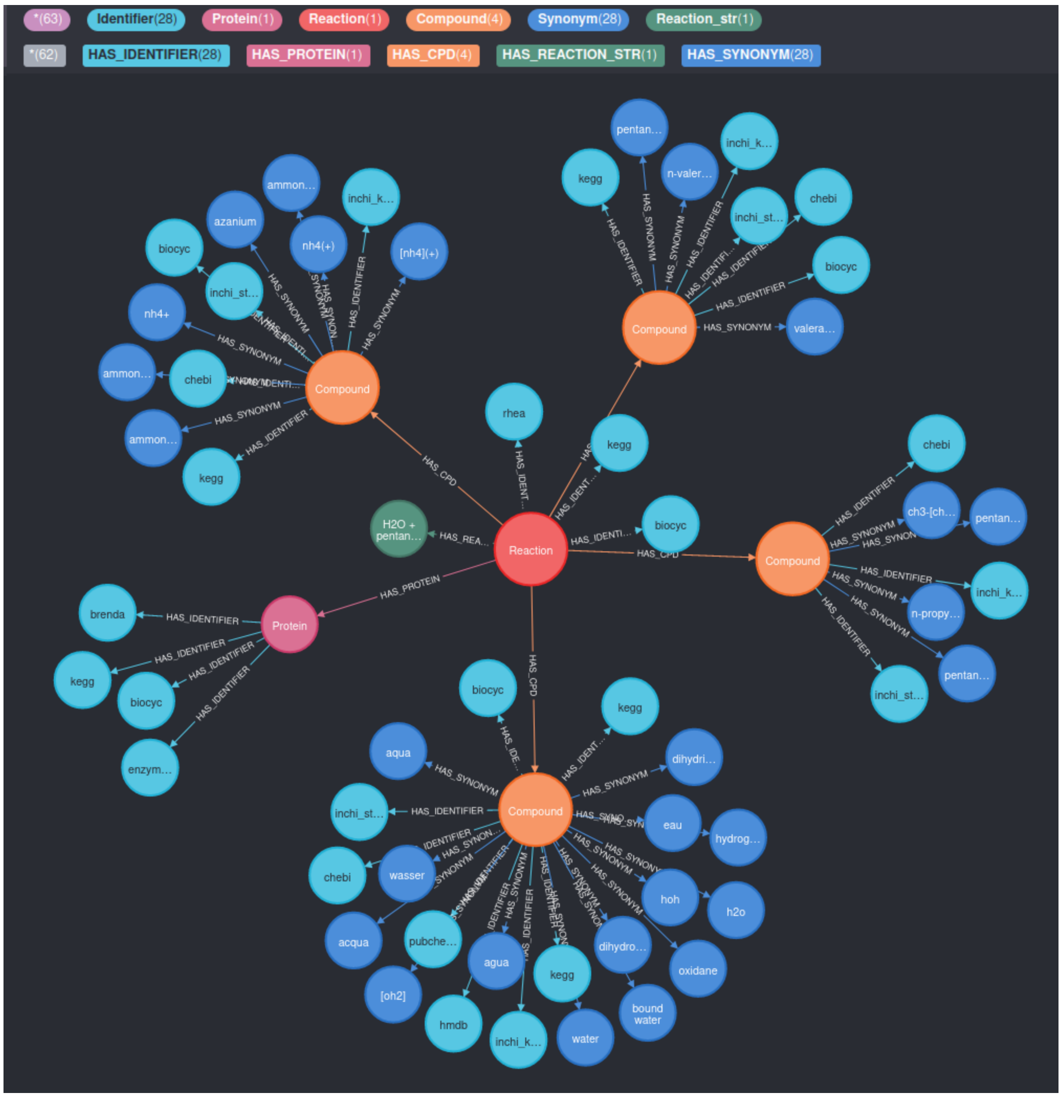


Figure 16 **Neo4j database** - this figure shows how UniFuncNet's data is represented in Neo4j. Now how each node main node (protein, reaction, and compound) is connected to sub-nodes (e.g., Identifiers and Synonyms). In this figure the command used was ”match (r:reaction) with r limit 1 match (i)–(p:Protein)–(r)–(c:Compound)–(j) where not i:Reaction and not j:Reaction) with i,p,r,c,j match (r)–(k) return i,p,r,c,j,k”, which matches with one reaction node, and then with the connecting protein and compound nodes as well as their respective annotations.

apply topological analysis and use meta-transcriptomics and meta-proteomics for the identification of keystone nodes [185] or differential analysis [5].

Another possibility would be to use the Neo4j database to aid in binning refinement. Assembly can be defined as the problem of reconstructing a contig (a contig represents a set of overlapping DNA) from the set of its k-mers/substrings (derived from reads). Binning refers to the clustering of assembled contigs to form MAGs. Current binning tools commonly use sequence features such as kmer-frequency, GC content, and read depth [86]. These features and associated methodologies vary depending on the underlying sequence technology (e.g., short vs long-read sequencing). Since metabolic information can be extracted from the binned and unbinned contigs, we attempted to use metabolic connectivity to assign unbinned contigs to bins. This work was based on a previously published proof of concept [15], that discussed how contigs could be potentially binned by mapping these to gaps in a metabolic network. To do so contig and bin-specific metabolic networks are generated and we then try to assign contigs based on whether any of their respective reactions fill any gaps in the bins' metabolic networks. Technically, this is done by checking the substrate(s) and product(s) of each reaction within the contig-specific metabolic network and then checking if the same substrates and products are present in any of the bins' metabolic network reactions. This can be scored via the metabolite connectivity score (MCS):

$$MCS_{ij} = \frac{|RS_i \cap NS_j|}{|RS_i|} + \frac{|RP_i \cap NP_j|}{|RP_i|} \tag{3}$$

where RS_i and RP_i are the set of substrates and products for reaction i, respectively; and NS_j and NP_j are the set of compounds not consumed or produced by any reaction in network j, respectively. In addition, to the MCS, we also used the following features to identify putative contigs assignments: paired-end reads information, taxonomy, kmer frequency, and read depth. Unfortunately, this project was discontinued since the refinement resulted in higher contamination (i.e., contigs assigned to the wrong bins) with negligible improvements in completeness (contigs assigned to the correct bins). We hypothesise this was due to the following reasons: poor or nonexistent contigs functional annotations, functional potential redundancy, the fact that most enzymatic reactions are reversible (which affects the NS_j and NP_j), the existence of very common cofactors (to address this we tried to scale cofactors using a method similar to TF-IDF). However, we found that the use of paired-end reads information led to minor improvements in the bins, therefore this information will be added to the in-house developed binning tool - binny.

Overall, this dynamically generated Neo4j database could provide a flexible framework in many different scenarios. By providing an API to generate (i.e., using UniFuncNet to integrate data from multiple databases) and interact with this network database, additional workflows could be built on top of it, depending on the users end goal.

8 Conclusion

In this thesis, I have shown how the different software developed during my PhD can address the challenge of integrating biological data in a scalable manner but also able to deal with the requirements of different types of users. Thematically, this thesis is mostly related to function annotation, however, I believe my main contribution towards the field lies in the ability to integrate data from multiple sources into a composite framework, be it in relation to protein function or network annotation. The tools developed offer significant customisation and can be used in multiple scenarios; emphasising versatility in software development is important, as it reduces the waste of human expertise that would otherwise be spent on developing partially redundant software. In this regard, I have shown how these tools are being applied by the community they were built for, but also how they can be used in different scenarios (e.g., Mantis is being used for different end-goals, i.e., functional annotation, binning, metabolic modelling).

I hope this thesis inspires future PhD students to develop their own high-quality software. Bioinformatics tools provide the foundation of many great works, accordingly, it should not be an afterthought in Science. While I expect the tools I implemented may one day be surpassed by better tools, I hope they can be useful to the community for years to come. Developing user-friendly, customizable, and scalable tools, should not have to be a naive dream.

Acronyms

API application programming interface.

AUC area under the ROC curve.

BPO best prediction only.

CDEM newly connected dead end metabolites.

CE combination e-value.

cID common database identifier.

DEM dead end metabolites.

DFS depth first search.

DNA deoxyriboNucleic acid.

EC enzyme commission identifier.

ERC European Research Council.

ETL extract, transform, load.

FAIR findability, accessibility, interoperability, and reuse.

FN false negative.

FP false positive.

GFS glacier fed stream sediment.

GO gene ontology identifier.

GSMM genome-scale metabolic model.

HC average hit coverage.

HCN average hit consistency.

HMLN heterogeneous multi-layered network.

HMM hidden Markov model.

HMMW reference hidden Markov model weight.

ID identifier.

KO KEGG orthology identifier.

MAG metagenome-assembled genomes.

MCS metabolite connectivity score.

MQ metadata quality.

MSA multiple sequence alignment.

NLP natural language processing.

NP_j set of compounds not produced by any reaction in network j.

NPFM NCBI protein family models.

NS_j set of compounds not consumed by any reaction in network j.

PFA protein function annotation.

PHMM profile hidden Markov model.

pID possible database identifier.

PNA potentially new annotation.

PoST	part-of-speech tagging.
RLC	reactions in the largest network component.
RP_i	set of products for reaction i.
RS_i	set of substrates for reaction i.
SIF	simple interaction format.
SS	similarity score.
TC	total coverage.
TES	token encoding and scoring.
TF-IDF	term frequency-inverse document frequency.
TN	true negative.
TP	true positive.
TSA	taxa-specific annotation.
TSHMM	taxon-specific hidden Markov model.
VMH	virtual metabolic human.

References

[1] Md Zahangir Alom et al. “A state-of-the-art survey on deep learning theory and architectures”. In: *Electronics* 8.3 (2019), p. 292.

[2] Hend Alrasheed. “Word synonym relationships for text analysis: A graph-based approach”. In: *Plos one* 16.7 (2021), e0255127.

[3] Stephen F Altschul et al. “Basic local alignment search tool”. In: *Journal of molecular biology* 215.3 (1990), pp. 403–410.

[4] Stephen F. Altschul et al. “Basic local alignment search tool”. In: *Journal of Molecular Biology* 215.3 (1990), pp. 403–410. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2.

[5] Victória Pascal Andreu et al. “BiG-MAP: an automated pipeline to profile metabolic gene cluster abundance and expression in microbiomes”. In: *mSystems* 6.5 (2021).

[6] Takuya Aramaki et al. “KofamKOALA: KEGG Ortholog assignment based on profile HMM and adaptive score threshold”. In: *Bioinformatics* 36.7 (2020), pp. 2251–2252. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz859. (Visited on 06/25/2020).

[7] Fabricio Almeida Araujo et al. “GO FEAT: a rapid web-based functional annotation tool for genomic and transcriptomic data”. In: *Scientific Reports* 8.1 (2018), p. 1794. ISSN: 2045-2322. DOI: 10.1038/s41598-018-20211-9.

[8] Carolina Arias et al. “KSHV 2.0: A Comprehensive Annotation of the Kaposi’s Sarcoma-Associated Herpesvirus Genome Using Next-Generation Sequencing Reveals Novel Genomic and Functional Features”. In: *PLOS Pathogens* 10.1 (2014), e1003847. ISSN: 1553-7374. DOI: 10.1371/journal.ppat.1003847.

[9] Michael Ashburner et al. “Gene Ontology: tool for the unification of biology”. In: *Nature genetics* 25.1 (2000), pp. 25–29. ISSN: 1061-4036. DOI: 10.1038/75556.

[10] Ramy K. Aziz et al. “The RAST Server: Rapid Annotations using Subsystems Technology”. In: *BMC Genomics* 9.1 (2008), p. 75. ISSN: 1471-2164. DOI: 10.1186/1471-2164-9-75.

[11] Parit Bansal et al. “Rhea, the reaction knowledgebase in 2022”. In: *Nucleic acids research* (2021).

[12] Stefan Behnel et al. “Cython: The Best of Both Worlds”. In: *Computing in Science Engineering* 13.2 (2011), pp. 31–39. ISSN: 1558-366X. DOI: 10.1109/MCSE.2010.118.

[13] Sidahmed Benabderrahmane et al. “IntelliGO: a new vector-based semantic similarity measure including annotation origin”. In: *BMC Bioinformatics* 11 (2010), p. 588. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-588.

[14] Helen M Berman et al. “The protein data bank”. In: *Nucleic acids research* 28.1 (2000), pp. 235–242.

[15] Matthew B. Biggs and Jason A. Papin. “Metabolic network-guided binning of metagenomic sequence fragments”. In: *Bioinformatics* 32.6 (Nov. 2015), pp. 867–874. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv671. eprint: https://academic.oup.com/bioinformatics/article-pdf/32/6/867/32742932/btv671.pdf. URL: https://doi.org/10.1093/bioinformatics/btv671.

[16] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 2009. URL: https://www.nltk.org/book/.

[17] Bo-Christer Bjork and Timo Korkeamaki. “Adoption of the open access business model in scientific journal publishing: A cross-disciplinary study”. In: *arXiv preprint arXiv:2005.01008* (2020).

[18] Edik M Blais, Arvind K Chavali, and Jason A Papin. “Linking genome-scale metabolic modeling and genome annotation”. In: *Systems Metabolic Engineering*. Springer, 2013, pp. 61–83.

[19] Thomas Blumenthal. “Gene clusters and polycistronic transcription in eukaryotes”. In: *Bioessays* 20.6 (1998), pp. 480–487.

[20] Thomas Blumenthal. “Operons in eukaryotes”. In: *Briefings in Functional Genomics* 3.3 (2004), pp. 199–211.

[21] Debby Bogaert, Ronald de Groot, and PWM Hermans. “Streptococcus pneumoniae colonisation: the key to pneumococcal disease”. In: *The Lancet infectious diseases* 4.3 (2004), pp. 144–154.

[22] Piotr Bojanowski et al. “Enriching word vectors with subword information”. In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[23] Karsten M. Borgwardt et al. “Protein function prediction via graph kernels”. In: *Bioinformatics* 21 (suppl_1 2005), pp. i47–i56. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti1007.

[24] Götz Bosse, Jan-Philipp Breuer, and Claudia Spies. “The resistance to changing guidelines–what are the challenges and how to meet them”. In: *Best Practice & Research Clinical Anaesthesiology* 20.3 (2006), pp. 379–395.

[25] C. Titus Brown and Luiz Irber. “sourmash: a library for MinHash sketching of DNA”. In: *J. Open Source Softw.* 1.5 (2016), p. 27.

[26] Benjamin Buchfink, Chao Xie, and Daniel H. Huson. “Fast and sensitive protein alignment using DIAMOND”. In: *Nature Methods* 12.1 (2015), pp. 59–60. ISSN: 1548-7105. DOI: 10.1038/nmeth.3176.

[27] Robin Buell et al. *Breaking the Bottleneck of Genomes: Understanding Gene Function Across Taxa*. US Department of Energy, Office of Biological and Environmental Research, 2018, p. 72.

[28] Susheel Bhanu Busi et al. “Optimised biomolecular extraction for metagenomic analysis of microbial biofilms from high-mountain streams”. In: *PeerJ* 8 (Oct. 2020), e9973. ISSN: 2167-8359. DOI: 10.7717/peerj.9973. URL: https://doi.org/10.7717/peerj.9973.

[29] Tunahan Çakır and Mohammad Jafar Khatibipour. “Metabolic network discovery by top-down and bottom-up approaches and paths for reconciliation”. In: *Frontiers in Bioengineering and Biotechnology* 2 (2014), p. 62.

[30] Ron Caspi et al. “The MetaCyc database of metabolic pathways and enzymes - a 2019 update”. In: *Nucleic Acids Res.* 48 (2019), pp. 445–453.

[31] Ron Caspi et al. “The MetaCyc database of metabolic pathways and enzymes-a 2019 update”. In: *Nucleic acids research* 48.D1 (2020), pp. D445–D453.

[32] Claudine Chaouiya. “Petri net modelling of biological networks”. In: *Briefings in bioinformatics* 8.4 (2007), pp. 210–219.

[33] Agnès Chapel et al. “An Extended Proteome Map of the Lysosomal Membrane Reveals Novel Potential Transporters”. In: *Molecular & Cellular Proteomics* 12.6 (2013), pp. 1572–1588. ISSN: 1535-9476, 1535-9484. DOI: `10.1074/mcp.M112.021980`.

[34] Francois Chollet et al. *Keras*. 2015. URL: `https://github.com/fchollet/keras`.

[35] Peter Cimermancic et al. “Insights into secondary metabolism from a global analysis of prokaryotic biosynthetic gene clusters”. In: *Cell* 158.2 (2014), pp. 412–421.

[36] Gene Ontology Consortium. “The gene ontology resource: 20 years and still GOing strong”. In: *Nucleic acids research* 47.D1 (2019), pp. D330–D338.

[37] UniProt Consortium. “UniProt: a worldwide hub of protein knowledge”. In: *Nucleic acids research* 47.D1 (2019), pp. D506–D515.

[38] UniProt Consortium. “UniProt: a worldwide hub of protein knowledge”. In: *Nucleic acids research* 47.D1 (2019), pp. D506–D515.

[39] Charles E Cook et al. “The European Bioinformatics Institute in 2016: data growth and integration”. In: *Nucleic acids research* 44.D1 (2016), pp. D20–D26.

[40] Charles E Cook et al. “The European Bioinformatics Institute in 2020: building a global infrastructure of interconnected data resources for the life sciences”. In: *Nucleic acids research* 48.D1 (2020), pp. D17–D23.

[41] NCBI Resource Coordinators. “Database resources of the National Center for Biotechnology Information”. In: *Nucleic Acids Res.* 46 (2017), pp. 8–13.

[42] Athel Cornish-Bowden. “Current IUBMB recommendations on enzyme nomenclature and kinetics”. In: *Perspectives in Science* 1.1 (2014), pp. 74–87.

[43] Nikolai Daraselia et al. “Automatic extraction of gene ontology annotation and its correlation with clusters in protein networks”. In: *BMC bioinformatics* 8 (2007), p. 243. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-243.

[44] Francesco Delogu. *fdelogu/SEM1b-Multiomics*. 2019.

[45] Lei Deng et al. “MADOKA: an ultra-fast approach for large-scale protein structure similarity searching”. In: *BMC Bioinformatics* 20.19 (2019), p. 662. ISSN: 1471-2105. DOI: 10.1186/s12859-019-3235-1.

[46] Paolo Di Tommaso et al. “Nextflow enables reproducible computational workflows”. In: *Nature biotechnology* 35.4 (2017), pp. 316–319.

[47] Oscar Dias et al. “Reconstructing genome-scale metabolic models with merlin”. In: *Nucleic acids research* 43.8 (2015), pp. 3899–3910.

[48] Robert C Edgar. “MUSCLE: a multiple sequence alignment method with reduced time and space complexity”. In: *BMC Bioinform.* 5.1 (2004), pp. 1–19.

[49] Rezvan Ehsani and Finn Drabløs. “TopoICSim: a new semantic similarity measure based on gene ontology”. In: *BMC bioinformatics* 17.1 (2016), p. 296. ISSN: 1471-2105. DOI: 10.1186/s12859-016-1160-0.

[50] Diana Ekman et al. “Multi-domain Proteins in the Three Kingdoms of Life: Orphan Domains and Other Unassigned Regions”. In: *Journal of Molecular Biology* 348.1 (2005), pp. 231–243. ISSN: 0022-2836. DOI: 10.1016/j.jmb.2005.02.007.

[51] Kenneth W Ellens et al. “Confronting the catalytic dark matter encoded by sequenced genomes”. In: *Nucleic acids research* 45.20 (2017), pp. 11495–11514.

[52] Kenneth W. Ellens et al. “Confronting the catalytic dark matter encoded by sequenced genomes”. In: *Nucleic Acids Research* 45.20 (2017), pp. 11495–11514. ISSN: 1362-4962. DOI: 10.1093/nar/gkx937.

[53] David M Emms and Steven Kelly. “OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy”. In: *Genome biology* 16.1 (2015), pp. 1–14.

[54] A Murat Eren et al. “Community-led, integrated, reproducible multi-omics with anvi’o”. In: *Nature microbiology* 6.1 (2021), pp. 3–6.

[55] Commission European. *Open Research Europe*. URL: https://open-research-europe.ec.europa.eu/.

[56] José P Faria et al. “Methods for automated genome-scale metabolic model reconstruction”. In: *Biochemical Society Transactions* 46.4 (2018), pp. 931–936.

[57] Adam M Feist et al. “Reconstruction of biochemical networks in microorganisms”. In: *Nature Reviews Microbiology* 7.2 (2009), pp. 129–143.

[58] Iddo Friedberg. “Automated protein function prediction—the genomic challenge”. In: *Briefings in bioinformatics* 7.3 (2006), pp. 225–242.

[59] Iddo Friedberg. “Automated protein function prediction—the genomic challenge”. In: *Briefings in Bioinformatics* 7.3 (2006), pp. 225–242. ISSN: 1467-5463. DOI: 10.1093/bib/bbl004.

[60] Kazuhiro A Fujita et al. “Integrating pathways of Parkinson’s disease in a molecular interaction map”. In: *Molecular neurobiology* 49.1 (2014), pp. 88–102.

[61] Sonali Vijay Gaikwad, Archana Chaugule, and Pramod Patil. “Text mining methods and techniques”. In: *International Journal of Computer Applications* 85.17 (2014).

[62] Jing Gao et al. “Metscape: a Cytoscape plug-in for visualizing and interpreting metabolomic data in the context of human metabolic networks”. In: *Bioinformatics* 26.7 (2010), pp. 971–973.

[63] Leyla Garcia et al. “FAIR adoption, assessment and challenges at UniProt”. In: *Scientific data* 6.1 (2019), pp. 1–4.

[64] Beatriz García-Jiménez, Jesús Torres-Bacete, and Juan Nogales. “Metabolic modelling approaches for describing and engineering microbial communities”. In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 226–246.

[65] Paul P Gardner et al. “Sustained software development, not number of citations or journal choice, is indicative of accurate bioinformatic software”. In: *bioRxiv* (2021), p. 092205.

[66] Jeff Gauthier et al. “A brief history of bioinformatics”. In: *Briefings in bioinformatics* 20.6 (2019), pp. 1981–1996.

[67] Sara El-Gebali et al. “The Pfam protein families database in 2019”. In: *Nucleic Acids Res.* 47 (2019), pp. 427–432.

[68] Molly K Gibson, Kevin J Forsberg, and Gautam Dantas. “Improved annotation of antibiotic resistance determinants reveals microbial resistomes cluster by ecology”. In: *ISME J* 9.1 (2015), pp. 207–216.

[69] Allison F Gillaspy et al. “The Staphylococcus aureus NCTC 8325 genome”. In: *Gram-Positive Pathogens* (2006), pp. 381–412.

[70] Stephen J Giovannoni et al. “Genome streamlining in a cosmopolitan oceanic bacterium”. In: *science* 309.5738 (2005), pp. 1242–1245.

[71] Vladimir Gligorijević and Nataša Pržulj. “Methods for biological data integration: perspectives and challenges”. In: *Journal of the Royal Society Interface* 12.112 (2015), p. 20150571.

[72] Carole Goble and Robert Stevens. “State of the nation in data integration for bioinformatics”. In: *Journal of biomedical informatics* 41.5 (2008), pp. 687–693.

[73] Willi Gottstein et al. “Constraint-based stoichiometric modelling from single organisms to microbial communities”. In: *Journal of the Royal Society Interface* 13.124 (2016), p. 20160627.

[74] Jason Grealey et al. “The carbon footprint of bioinformatics”. In: *BioRxiv* (2021).

[75] Marc Griesemer et al. “Combining multiple functional annotation tools increases coverage of metabolic annotation”. In: *BMC genomics* 19.1 (2018), pp. 1–11.

[76] Daniel H. Haft et al. “TIGRFAMs and Genome Properties in 2013”. In: *Nucleic Acids Research* 41 (Database issue 2013), pp. D387–D395. ISSN: 0305-1048. DOI: 10.1093/nar/gks1234.

[77] Geoffrey D Hannigan et al. “A deep learning genome-mining strategy for biosynthetic gene cluster prediction”. In: *Nucleic Acids Research* 47.18 (Aug. 2019), e110–e110. ISSN: 0305-1048. DOI: 10.1093/nar/gkz654.

[78] Zellig S Harris. “Distributional structure”. In: *Word* 10.2-3 (1954), pp. 146–162.

[79] Jennifer Harrow et al. “ELIXIR-EXCELERATE: establishing Europe’s data infrastructure for the life science research of the future”. In: *The EMBO Journal* 40.6 (2021), e107409.

[80] Janna Hastings et al. “ChEBI in 2016: Improved services and an expanding collection of metabolites”. In: *Nucleic acids research* 44.D1 (2016), pp. D1214–D1219.

[81] Kenneth Haug et al. “MetaboLights: a resource evolving in response to the needs of its scientific community”. In: *Nucleic acids research* 48.D1 (2020), pp. D440–D444.

[82] Almut Heinken et al. “DEMETER: efficient simultaneous curation of genome-scale reconstructions guided by experimental data and refined gene annotations”. In: *Bioinformatics* 37.21 (2021), pp. 3974–3975.

[83] Anna Heintz-Buschart et al. “Integrated multi-omics of the human gut microbiome in a case study of familial type 1 diabetes”. In: *Nature Microbiology* 2.1 (2016). Number: 1 Publisher: Nature Publishing Group, pp. 1–13. ISSN: 2058-5276. DOI: 10.1038/nmicrobiol.2016.180.

[84] Hauke Hennecke. “Nitrogen fixation genes involved in the Bradyrhizobium japonicum-soybean symbiosis”. In: *FEBS letters* 268.2 (1990), pp. 422–426.

[85] Rafael Hernández-de-Diego et al. “PaintOmics 3: a web resource for the pathway analysis and visualization of multi-omics data”. In: *Nucleic acids research* 46.W1 (2018), W503–W509.

[86] Oskar Hickl et al. “binny: an automated binning algorithm to recover high-quality genomes from complex metagenomic datasets”. In: *bioRxiv* (2021).

[87] Daniel S Himmelstein and Sergio E Baranzini. “Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes”. In: *PLoS computational biology* 11.7 (2015), e1004259.

[88] Paulien Hogeweg. “The roots of bioinformatics in theoretical biology”. In: *PLoS computational biology* 7.3 (2011), e1002021.

[89] Matthew Honnibal. *A Good Part-of-Speech Tagger in about 200 Lines of Python*. 2013. URL: `https://explosion.ai/blog/part-of-speech-pos-tagger-in-python`.

[90] Matthew Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: `10.5281/zenodo.1212303`.

[91] JoAnn Hoskins et al. “Genome of the bacterium Streptococcus pneumoniae strain R6”. In: *Journal of bacteriology* 183.19 (2001), pp. 5709–5717.

[92] Chung-Chi Huang and Zhiyong Lu. “Community challenges in biomedical text mining over 10 years: success, failure and the future”. In: *Briefings in Bioinformatics* 17.1 (2016), pp. 132–144. ISSN: 1477-4054. DOI: `10.1093/bib/bbv024`.

[93] Yue Huang, Mingxin Gan, and Rui Jiang. “Ontology-Based Genes Similarity Calculation with TF-IDF”. In: *LNCS* 7473 (2012), pp. 600–607.

[94] Jaime Huerta-Cepas et al. “eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses”. In: *Nucleic Acids Res.* 47 (2018), pp. 309–314.

[95] Jaime Huerta-Cepas et al. “eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses”. In: *Nucleic Acids Research* 47 (D1 2019), pp. D309–D314. ISSN: 0305-1048. DOI: `10.1093/nar/gky1085`.

[96] Jaime Huerta-Cepas et al. “Fast Genome-Wide Functional Annotation through Orthology Assignment by eggNOG-Mapper”. In: *Molecular Biology and Evolution* 34.8 (2017), pp. 2115–2122. ISSN: 0737-4038. DOI: 10.1093/molbev/msx148.

[97] Martijn Huynen et al. “Predicting protein function by genomic context: quantitative evaluation and qualitative inferences”. In: *Genome research* 10.8 (2000), pp. 1204–1210.

[98] Doug Hyatt et al. “Prodigal: prokaryotic gene recognition and translation initiation site identification”. In: *BMC Bioinformatics* 11 (2010), p. 119. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-119.

[99] Massimo Iorizzo et al. “De novo assembly and characterization of the carrot transcriptome reveals novel genes, new markers, and genetic diversity”. In: *BMC Genomics* 12.1 (2011), p. 389. ISSN: 1471-2164. DOI: 10.1186/1471-2164-12-389.

[100] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. “Facing imbalanced data–recommendations for the use of performance metrics”. In: *2013 Humaine association conference on affective computing and intelligent interaction*. IEEE. 2013, pp. 245–251.

[101] Philip Jones et al. “InterProScan 5: genome-scale protein function classification”. In: *Bioinformatics* 30.9 (2014), pp. 1236–1240. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu031.

[102] Philip Jones et al. “InterProScan 5: genome-scale protein function classification”. In: *Bioinformatics* 30.9 (2014), pp. 1236–1240.

[103] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[104] Minoru Kanehisa and Susumu Goto. “KEGG: Kyoto Encyclopedia of Genes and Genomes”. In: *Nucleic Acids Res.* 28 (2000), pp. 27–30.

[105] Minoru Kanehisa and Susumu Goto. “KEGG: kyoto encyclopedia of genes and genomes”. In: *Nucleic acids research* 28.1 (2000), pp. 27–30.

[106] Navneet Kaur and Deepak Garg. “Analysis of the Depth First Search Algorithms”. In: *Data mining and knowledge engineering* 4 (2012), pp. 37–41.

[107] Satria A Kautsar et al. “MIBiG 2.0: a repository for biosynthetic gene clusters of known function”. In: *Nucleic acids research* 48.D1 (2020), pp. D454–D458.

[108] Kevin P. Keegan, Elizabeth M. Glass, and Folker Meyer. “MG-RAST, a Metagenomics Service for Analysis of Microbial Community Structure and Function”. In: *Methods in Molecular Biology (Clifton, N.J.)* 1399 (2016), pp. 207–233. ISSN: 1940-6029. DOI: 10.1007/978-1-4939-3369-3_13.

[109] Sunghwan Kim et al. “PubChem substance and compound databases”. In: *Nucleic acids research* 44.D1 (2016), pp. D1202–D1213.

[110] Zachary A King et al. “BiGG Models: A platform for integrating, standardizing and sharing genome-scale models”. In: *Nucleic acids research* 44.D1 (2016), pp. D515–D522.

[111] William Klimke et al. “Solving the Problem: Genome Annotation Standards before the Data Deluge”. In: *Standards in Genomic Sciences* 5.1 (2011), pp. 168–193. ISSN: 1944-3277. DOI: 10.4056/sigs.2084864.

[112] Jan O Korbel et al. “Analysis of genomic context: prediction of functional associations from conserved bidirectionally transcribed gene pairs”. In: *Nature biotechnology* 22.7 (2004), pp. 911–917.

[113] Nicole M Koropatkin, Elizabeth A Cameron, and Eric C Martens. “How glycan metabolism shapes the human gut microbiota”. In: *Nature Reviews Microbiology* 10.5 (2012), pp. 323–335.

[114] Masaaki Kotera and Susumu Goto. “Metabolic pathway reconstruction strategies for central metabolism and natural product biosynthesis”. In: *Biophysics and physicobiology* 13 (2016), pp. 195–205.

[115] Mikaela Koutrouli et al. “A guide to conquer the biological network era using graph theory”. In: *Frontiers in bioengineering and biotechnology* 8 (2020), p. 34.

[116] Michael Kramer et al. “Inferring gene ontologies from pairwise similarity data”. In: *Bioinformatics (Oxford, England)* 30.12 (2014), pp. i34–42. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btu282.

[117] Michal Krassowski et al. “State of the field in multi-omics research: From computational needs to data mining and sharing”. In: *Frontiers in Genetics* 11 (2020).

[118] Sudhir Kumar, Glen Stecher, and Koichiro Tamura. “MEGA7: molecular evolutionary genetics analysis version 7.0 for bigger datasets”. In: *Molecular biology and evolution* 33.7 (2016), pp. 1870–1874.

[119] Benoit J. Kunath et al. “From proteins to polysaccharides: lifestyle and genetic evolution of Coprothermobacter proteolyticus”. In: *The ISME Journal* 13.3 (2019), pp. 603–617. ISSN: 1751-7370. DOI: 10.1038/s41396-018-0290-y.

[120] Tien-Chueh Kuo, Tze-Feng Tian, and Yufeng Jane Tseng. “3Omics: a web-based systems biology tool for analysis, integration and visualization of human transcriptomic, proteomic and metabolomic data”. In: *BMC systems biology* 7.1 (2013), pp. 1–15.

[121] Vasileios Lapatas et al. “Data integration in biological research: an overview”. In: *J Biol Res (Thessalon)* 22.1 (2015). DOI: 10.1186/s40709-015-0032-5.

[122] Roman A Laskowski, James D Watson, and Janet M Thornton. “ProFunc: a server for predicting protein function from 3D structure”. In: *Nucleic acids research* 33.suppl_2 (2005), W89–W93.

[123] Oluwasola Lawal et al. “Headspace volatile organic compounds from bacteria implicated in ventilator-associated pneumonia analysed by TD-GC/MS”. In: *Journal of breath research* 12.2 (2018), p. 026002.

[124] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[125] Bohyun Lee et al. “Heterogeneous multi-layered network model for omics data integration and analysis”. In: *Frontiers in genetics* 10 (2020), p. 1381.

[126] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

[127] Jonathan G. Lees et al. “Gene3D: Multi-domain annotations for protein sequence and comparative genome analysis”. In: *Nucleic Acids Research* 42 (Database issue 2014), pp. D240–D245. ISSN: 0305-1048. DOI: 10.1093/nar/gkt1205.

[128] Meng Liu and Paul D. Thomas. “GO functional similarity clustering depends on similarity measure, clustering method, and annotation completeness”. In: *BMC bioinformatics* 20.1 (2019), p. 155. ISSN: 1471-2105. DOI: 10.1186/s12859-019-2752-2.

[129] Yaniv Loewenstein et al. “Protein function annotation by homology-based inference”. In: *Genome Biol.* 10.2 (2009), pp. 1–8.

[130] Marc Lohse et al. “Mercator: a fast and simple web server for genome scale functional annotation of plant sequence data”. In: *Plant, Cell & Environment* 37.5 (2014), pp. 1250–1258. ISSN: 1365-3040. DOI: 10.1111/pce.12231.

[131] Shennan Lu et al. “CDD/SPARCLE: the conserved domain database in 2020”. In: *Nucleic acids research* 48.D1 (2020), pp. D265–D268.

[132] Daniel Machado et al. “Fast automated reconstruction of genome-scale metabolic models for microbial species and communities”. In: *Nucleic acids research* 46.15 (2018), pp. 7542–7553.

[133] Amanda Mackie et al. “Dead end metabolites-defining the known unknowns of the E. coli metabolic network”. In: *PloS one* 8.9 (2013), e75210.

[134] Stefanía Magnúsdóttir et al. “Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota”. In: *Nature biotechnology* 35.1 (2017), pp. 81–89.

[135] Serghei Mangul et al. “Challenges and recommendations to improve the installability and archival stability of omics computational tools”. In: *PLOS Biology* 17.6 (June 2019), pp. 1–16. DOI: 10.1371/journal.pbio.3000333. URL: https://doi.org/10.1371/journal.pbio.3000333.

[136] Serghei Mangul et al. "Challenges and recommendations to improve the installability and archival stability of omics computational tools". In: *PLoS biology* 17.6 (2019), e3000333.

[137] Serghei Mangul et al. *Improving the usability and archival stability of bioinformatics software*. 2019.

[138] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[139] Olivia U. Mason et al. "Metagenomics reveals sediment microbial community response to Deepwater Horizon oil spill". In: *The ISME Journal* 8.7 (2014), pp. 1464–1475. ISSN: 1751-7370. DOI: `10.1038/ismej.2013.254`.

[140] Konstantinos Mavromatis et al. "Gene Context Analysis in the Integrated Microbial Genomes (IMG) Data Management System". In: *PLOS ONE* 4.11 (2009), e7979. ISSN: 1932-6203. DOI: `10.1371/journal.pone.0007979`.

[141] Michael McClelland et al. "Complete genome sequence of Salmonella enterica serovar Typhimurium LT2". In: *Nature* 413.6858 (2001), pp. 852–856.

[142] Simon Jon McIlroy et al. "Metabolic model for the filamentous 'Candidatus Microthrix parvicella'based on genomic and metagenomic analyses". In: *The ISME journal* 7.6 (2013), pp. 1161–1172.

[143] Sebastián N Mendoza et al. "A systematic assessment of current genome-scale metabolic reconstruction tools". In: *Genome biology* 20.1 (2019), pp. 1–20.

[144] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv* (2013).

[145] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41.

[146] Alex L Mitchell et al. "MGnify: the microbiome analysis resource in 2020". In: *Nucleic Acids Res.* 48 (2019), pp. 570–578.

[147] Alex L. Mitchell et al. “MGnify: the microbiome analysis resource in 2020”. In: *Nucleic Acids Research* 48 (D1 2020), pp. D570–D578. ISSN: 0305-1048. DOI: 10.1093/nar/gkz1035.

[148] Felix Mölder et al. “Sustainable data analysis with Snakemake”. In: *F1000Research* 10 (2021).

[149] Michael A. Mooney et al. “Functional and Genomic Context in Pathway Analysis of GWAS Data”. In: *Trends in genetics : TIG* 30.9 (2014), pp. 390–400. ISSN: 0168-9525. DOI: 10.1016/j.tig.2014.07.004.

[150] Sébastien Moretti et al. “MetaNetX/MNXref: Unified namespace for metabolites and biochemical reactions in the context of metabolic models”. In: *Nucleic Acids Research* 49.D1 (2021), pp. D570–D574.

[151] Emilie Muller et al. “Condensing the omics fog of microbial communities”. In: *Trends in microbiology* 21 (June 2013). DOI: 10.1016/j.tim.2013.04.009.

[152] Emilie EL Muller et al. “Using metabolic networks to resolve ecological properties of microbiomes”. In: *Current Opinion in Systems Biology* 8 (2018), pp. 73–80.

[153] Shaman Narayanasamy et al. “IMP: a pipeline for reproducible reference-independent integrated metagenomic and metatranscriptomic analyses”. In: *Genome biology* 17.1 (2016), pp. 1–21.

[154] Mark Neumann et al. *ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing*. ACL, 2019, pp. 319–327.

[155] Felicia N. New and Ilana L. Brito. “What Is Metagenomics Teaching Us, and What Is Missed?” eng. In: *Annual Review of Microbiology* 74 (Sept. 2020), pp. 117–135. ISSN: 1545-3251. DOI: 10.1146/annurev-micro-012520-072314.

[156] Nam D Nguyen and Daifeng Wang. “Multiview learning for understanding functional multiomics”. In: *PLoS computational biology* 16.4 (2020), e1007677.

[157] Zoran Nikoloski et al. “Metabolic networks are NP-hard to reconstruct”. In: *Journal of theoretical biology* 254.4 (2008), pp. 807–816.

[158] Henrik Nordberg et al. “The genome portal of the Department of Energy Joint Genome Institute: 2014 updates”. In: *Nucleic Acids Res.* 42 (2013), pp. 26–31.

[159] Alberto Noronha et al. “The Virtual Metabolic Human database: integrating human and gut microbiome metabolism with nutrition and disease”. In: *Nucleic acids research* 47.D1 (2019), pp. D614–D624.

[160] Jing Ouyang et al. “The bacterium Akkermansia muciniphila: a sentinel for gut permeability and its relevance to HIV-related inflammation”. In: *Frontiers in immunology* 11 (2020), p. 645.

[161] Ross Overbeek et al. “The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes”. In: *Nucleic Acids Research* 33.17 (2005), pp. 5691–5702. ISSN: 1362-4962. DOI: `10.1093/nar/gki866`.

[162] Donovan H Parks et al. “CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes”. In: *Genome Res.* 25.7 (2015), pp. 1043–1055.

[163] Donovan H Parks et al. “GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy”. In: *Nucleic Acids Res* (2021).

[164] Donovan H. Parks et al. “A complete domain-to-species taxonomy for Bacteria and Archaea”. In: *Nature Biotechnology* 38.9 (2020), pp. 1079–1086. ISSN: 1546-1696. DOI: `10.1038/s41587-020-0501-8`.

[165] Donovan H. Parks et al. “A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life”. In: *Nature Biotechnology* 36.10 (2018), pp. 996–1004. ISSN: 1546-1696. DOI: `10.1038/nbt.4229`.

[166] Edoardo Pasolli et al. “Extensive Unexplored Human Microbiome Diversity Revealed by Over 150,000 Genomes from Metagenomes Spanning Age, Geography, and Lifestyle”. In: *Cell* 176.3 (2019), 649–662.e20. ISSN: 0092-8674. DOI: `10.1016/j.cell.2019.01.001`.

[167] William R Pearson. “An introduction to sequence similarity (“homology”) searching”. In: *Current protocols in bioinformatics* 42.1 (2013), pp. 3–1.

[168] Jiajie Peng et al. “Measuring semantic similarities by combining gene ontology annotations and gene co-function networks”. In: *BMC bioinformatics* 16 (2015). ISSN: 1471-2105. DOI: 10.1186/s12859-015-0474-7.

[169] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. 2014, pp. 1532–1543.

[170] Catia Pesquita et al. “Semantic similarity in biomedical ontologies”. In: *PLoS computational biology* 5.7 (2009), e1000443. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000443.

[171] Friedhelm Pfeiffer and Dieter Oesterhelt. “A Manual Curation Strategy to Improve Genome Annotation: Application to a Set of Haloarchael Genomes”. In: *Life* 5.2 (2015), pp. 1427–1444. ISSN: 2075-1729. DOI: 10.3390/life5021427.

[172] Vasilis J. Promponas, Ioannis Iliopoulos, and Christos A. Ouzounis. “Annotation inconsistencies beyond sequence similarity-based function prediction – phylogeny and genome structure”. In: *Standards in Genomic Sciences* 10 (2015). ISSN: 1944-3277. DOI: 10.1186/s40793-015-0101-2.

[173] Michael E Pyne, Lauren Narcross, and Vincent JJ Martin. “Engineering plant secondary metabolism in microbial systems”. In: *Plant physiology* 179.3 (2019), pp. 844–861.

[174] Pedro Queiros et al. “UniFuncNet: a flexible network annotation framework”. In: *bioRxiv* (2022).

[175] Pedro Queirós. *Mantis - Wiki*. https://github.com/PedroMTQ/mantis/wiki. 2020.

[176] Pedro Queirós et al. “Mantis: flexible and consensus-driven genome annotation”. In: *GigaScience* 10.6 (June 2021). giab042. ISSN: 2047-217X. DOI: 10.1093/gigascience/giab042. eprint: https://academic.oup.com/gigascience/article-pdf/10/6/

giab042/38445405/giab042_reviewer_2_report_revision_1.pdf. URL: https://doi.org/10.1093/gigascience/giab042.

[177] Pedro Queirós et al. “Mantis: flexible and consensus-driven genome annotation”. In: *GigaScience* 10.6 (2021), giab042.

[178] Pedro Queirós et al. *Supporting data for ”Mantis: flexible and consensus-driven genome annotation”*. 2021. URL: http://dx.doi.org/10.5524/100903.

[179] Pedro Queirós et al. “Unification of functional annotation descriptions using text mining”. In: *Biological Chemistry* 402.8 (2021), pp. 983–990.

[180] Christopher Quince et al. “Shotgun metagenomics, from sampling to analysis”. eng. In: *Nature Biotechnology* 35.9 (Sept. 2017), pp. 833–844. ISSN: 1546-1696. DOI: 10.1038/nbt.3935.

[181] Leonard Richardson. “Beautiful soup documentation”. In: *April* (2007).

[182] Daniel J Rigden and Xosé M Fernández. “The 2021 Nucleic Acids Research database issue and the online molecular biology database collection”. In: *Nucleic Acids Research* 49.D1 (Dec. 2020), pp. D1–D9. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa1216. eprint: https://academic.oup.com/nar/article-pdf/49/D1/D1/35364664/gkaa1216.pdf. URL: https://doi.org/10.1093/nar/gkaa1216.

[183] Sean Roberts Eddy. *HMMER: biosequence analysis using profile hidden Markov models*. 2020. URL: http://hmmer.org/.

[184] Huigui Rong et al. “Optimizing energy consumption for data centers”. In: *Renewable and Sustainable Energy Reviews* 58 (2016), pp. 674–691.

[185] Hugo Roume et al. “Comparative integrated omics: identification of key functionalities in microbial community-wide metabolic networks”. In: *npj Biofilms and Microbiomes* 1.1 (2015), pp. 1–11.

[186] Jae Yong Ryu, Hyun Uk Kim, and Sang Yup Lee. “Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers”. In: *Proceedings of*

the National Academy of Sciences 116.28 (2019), pp. 13996–14001. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1821905116.

[187] Jr Saier Milton H., Can V. Tran, and Ravi D. Barabote. “TCDB: the Transporter Classification Database for membrane transport protein analyses and information”. In: *Nucleic Acids Res.* 34 (2006), pp. 181–186.

[188] Milton H Saier Jr et al. “The Transporter Classification Database (TCDB): 2021 update”. In: *Nucleic Acids Research* 49.D1 (2021), pp. D461–D467.

[189] Gayathri Sambamoorthy and Karthik Raman. “Understanding the evolution of functional redundancy in metabolic networks”. In: *Bioinformatics* 34.17 (2018), pp. i981–i987.

[190] Iqbal H Sarker. “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”. In: *SN Computer Science* 2.6 (2021), pp. 1–20.

[191] Jan Schellenberger et al. “BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions”. In: *BMC bioinformatics* 11.1 (2010), pp. 1–10.

[192] Kirstin Scherlach and Christian Hertweck. “Mining and unearthing hidden biosynthetic potential”. In: *Nature Communications* 12.1 (2021), pp. 1–12.

[193] Alexandra M. Schnoes et al. “Annotation Error in Public Databases: Misannotation of Molecular Function in Enzyme Superfamilies”. In: *PLoS Computational Biology* 5.12 (2009). ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.1000605.

[194] Torsten Seemann. “Prokka: rapid prokaryotic genome annotation”. In: *Bioinformatics* 30.14 (2014), pp. 2068–2069. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btu153. (Visited on 06/25/2020).

[195] Nicola Segata et al. “Computational meta’omics for microbial community studies”. In: *Molecular Systems Biology* 9 (May 14, 2013). ISSN: 1744-4292. DOI: 10.1038/msb.2013.22.

[196] Paul Shannon et al. “Cytoscape: a software environment for integrated models of biomolecular interaction networks”. In: *Genome research* 13.11 (2003), pp. 2498–2504.

[197] Abdul R Sheik et al. “In situ phenotypic heterogeneity among single cells of the filamentous bacterium Candidatus Microthrix parvicella”. In: *The ISME journal* 10.5 (2016), pp. 1274–1279.

[198] Dan Sholler et al. “Resistance to Adoption of Best Practices”. In: (2019).

[199] Fabian Sievers et al. “Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega”. In: *Mol. Syst. Biol.* 7 (2011). DOI: `10.1038/msb.2011.75`.

[200] Christian J. A. Sigrist et al. “New and continuing developments at PROSITE”. In: *Nucleic Acids Research* 41 (Database issue 2013), pp. D344–347. ISSN: 1362-4962. DOI: `10.1093/nar/gks1067`.

[201] Arthur Fernandes Siqueira et al. “Comparative genomics of Bradyrhizobium japonicum CPAC 15 and Bradyrhizobium diazoefficiens CPAC 7: elite model strains for understanding symbiotic performance with soybean”. In: *BMC genomics* 15.1 (2014), pp. 1–21.

[202] Luke T Slater et al. “Improved characterisation of clinical text through ontology-based vocabulary expansion”. In: *J Biomed Semant* (2021). DOI: `10.1186/s13326-021-00241-5`.

[203] “Standardizing data”. In: *Nature Cell Biology* 10.10 (2008), pp. 1123–1124. ISSN: 1476-4679. DOI: `10.1038/ncb1008-1123`.

[204] Lincoln Stein. “Genome annotation: from sequence to biology”. In: *Nat. Rev. Genet.* 2.7 (2001), pp. 493–503.

[205] Lincoln D Stein. “Integrating biological databases”. In: *Nat. Rev. Genet.* 4.5 (2003), pp. 337–345.

[206] Martin Steinegger and Johannes Söding. “MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets”. In: *Nat. Biotechnol.* 35.11 (2017), pp. 1026–1028.

[207] Martin Steinegger and Johannes Söding. “MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets”. In: *Nature biotechnology* 35.11 (2017), pp. 1026–1028.

[208] Martin Steinegger et al. “HH-suite3 for fast remote homology detection and deep protein annotation”. In: *BMC Bioinformatics* 20.1 (2019), p. 473. ISSN: 1471-2105. DOI: 10.1186/s12859-019-3019-7.

[209] Ralf Steuer et al. “Observing and interpreting correlations in metabolomic networks”. In: *Bioinformatics* 19.8 (2003), pp. 1019–1026.

[210] Robert Stevens et al. “Ontologies in bioinformatics”. In: *Handbook on ontologies*. Springer, 2004, pp. 635–657.

[211] C K. Stover et al. “Complete genome sequence of Pseudomonas aeruginosa PAO1, an opportunistic pathogen”. In: *Nature* 406.6799 (2000), pp. 959–964.

[212] Hong Su et al. “Improved Protein Structure Prediction Using a New Multi-Scale Network and Homologous Templates”. In: *Advanced Science* (2021), p. 2102592.

[213] Indhupriya Subramanian et al. “Multi-omics data integration, interpretation, and its application”. In: *Bioinformatics and biology insights* 14 (2020), p. 1177932219899051.

[214] Shinichi Sunagawa et al. “Structure and function of the global ocean microbiome”. In: *Science* 348.6237 (2015). Ed. by Emmanuel Boss et al. ISSN: 0036-8075. DOI: 10.1126/science.1261359. eprint: https://science.sciencemag.org/content/348/6237/1261359.full.pdf. URL: https://science.sciencemag.org/content/348/6237/1261359.

[215] Shinichi Sunagawa et al. “Structure and function of the global ocean microbiome”. In: *Science* 348.6237 (2015). Publisher: American Association for the Advancement

of Science Section: Research Article. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1261359.

[216] Ahmet Sureyya Rifaioglu et al. “DEEPred: Automated Protein Function Prediction with Multi-task Feed-forward Deep Neural Networks”. In: *Scientific Reports* 9.1 (2019), p. 7344. ISSN: 2045-2322. DOI: 10.1038/s41598-019-43708-3.

[217] Damian Szklarczyk et al. “STITCH 5: augmenting protein–chemical interaction networks with tissue and affinity data”. In: *Nucleic acids research* 44.D1 (2016), pp. D380–D384.

[218] Damian Szklarczyk et al. “STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets”. In: *Nucleic Acids Research* 47 (D1 2019), pp. D607–D613. ISSN: 1362-4962. DOI: 10.1093/nar/gky1131.

[219] Roman L. Tatusov et al. “The COG database: a tool for genome-scale analysis of protein functions and evolution”. In: *Nucleic Acids Res.* 28 (2000), pp. 33–36.

[220] Ann Taylor, Mitchell Marcus, and Beatrice Santorini. *The Penn Treebank: An Overview*. Ed. by Anne Abeillé. Springer Netherlands, 2003, pp. 5–22.

[221] Ines Thiele and Bernhard Ø Palsson. “A protocol for generating a high-quality genome-scale metabolic reconstruction”. In: *Nature protocols* 5.1 (2010), pp. 93–121.

[222] Paul Thompson et al. “The BioLexicon: a large-scale terminological resource for biomedical text mining”. In: *BMC Bioinformatics* 12.1 (2011), p. 397. ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-397.

[223] Liang Tian et al. “Deciphering functional redundancy in the human microbiome”. In: *Nature communications* 11.1 (2020), pp. 1–11.

[224] Michelle L. Treiber et al. “Pre- and post-sequencing recommendations for functional annotation of human fecal metagenomes”. In: *BMC Bioinformatics* 21.1 (2020), p. 74. ISSN: 1471-2105. DOI: 10.1186/s12859-020-3416-y.

[225] Sipko Van Dam et al. “Gene co-expression analysis for functional classification and gene–disease predictions”. In: *Briefings in bioinformatics* 19.4 (2018), pp. 575–592.

[226] Tim Van Den Bossche et al. “Critical Assessment of MetaProteome Investigation (CAMPI): a multi-laboratory comparison of established workflows”. In: *Nature communications* 12.1 (2021), pp. 1–15.

[227] Mark WJ Van Passel et al. “The genome of Akkermansia muciniphila, a dedicated intestinal mucin degrader, and its use in exploring intestinal metagenomes”. In: *PloS one* 6.3 (2011), e16876.

[228] Sébastien Varrette et al. “Management of an Academic HPC Cluster: The UL Experience”. In: (2014). URL: `https://hpc.uni.lu`.

[229] Alexei Vazquez et al. “Global protein function prediction from protein-protein interaction networks”. In: *Nature Biotechnology* 21.6 (2003), pp. 697–700. ISSN: 1546-1696. DOI: `10.1038/nbt825`.

[230] Karin M. Verspoor et al. “Text Mining Improves Prediction of Protein Functional Sites”. In: *PLOS ONE* 7.2 (2012), pp. 1–16.

[231] Sheng Wang et al. “Annotating gene sets by mining large literature collections with protein networks”. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 23 (2018), pp. 602–613. ISSN: 2335-6936.

[232] Leon Weber et al. “HunFlair: an easy-to-use tool for state-of-the-art biomedical named entity recognition”. In: *Bioinformatics* (2021). DOI: `10.1093/bioinformatics/btab042`.

[233] Rodney A Welch et al. “Extensive mosaic structure revealed by the complete genome sequence of uropathogenic Escherichia coli”. In: *Proceedings of the National Academy of Sciences* 99.26 (2002), pp. 17020–17024.

[234] James C Whisstock and Arthur M Lesk. “Prediction of protein function from protein sequence and structure”. In: *Quarterly reviews of biophysics* 36.3 (2003), pp. 307–340.

[235] James C. Whisstock and Arthur M. Lesk. “Prediction of protein function from protein sequence and structure”. In: *Quarterly Reviews of Biophysics* 36.3 (2003), pp. 307–340. ISSN: 0033-5835. DOI: 10.1017/s0033583503003901.

[236] Mark D. Wilkinson et al. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3.1 (2016), p. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18.

[237] David S Wishart et al. “HMDB 4.0: the human metabolome database for 2018”. In: *Nucleic acids research* 46.D1 (2018), pp. D608–D617.

[238] Michael Woelfle, Piero Olliaro, and Matthew H Todd. “Open science is a research accelerator”. In: *Nature chemistry* 3.10 (2011), pp. 745–748.

[239] Sitao Wu et al. “WebMGA: a customizable web server for fast metagenomic sequence analysis”. In: *BMC genomics* 12 (2011). ISSN: 1471-2164. DOI: 10.1186/1471-2164-12-444.

[240] Corin Yeats, Oliver C. Redfern, and Christine Orengo. “A fast and automated solution for accurately resolving protein domain architectures”. In: *Bioinformatics* 26.6 (2010), pp. 745–751. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btq034.

[241] JIA Yu and Joann K Whalen. “A new perspective on functional redundancy and phylogenetic niche conservatism in soil microbial communities”. In: *Pedosphere* 30.1 (2020), pp. 18–24.

[242] Zhiqiang Zeng et al. “Survey of Natural Language Processing Techniques in Bioinformatics”. In: *Computational and Mathematical Methods in Medicine* 2015 (2015), p. 674296. ISSN: 1748-6718. DOI: 10.1155/2015/674296.

[243] Zhang Zhang et al. “Data integration in bioinformatics: current efforts and challenges”. In: *Bioinformatics-Trends and Methodologies* (2011), pp. 41–56.

[244] Bihai Zhao et al. “An efficient method for protein function annotation based on multilayer protein networks”. In: *Human Genomics* 10 (2016). ISSN: 1473-9542. DOI: 10.1186/s40246-016-0087-x.

[245] Guangyan Zhou and Jianguo Xia. "OmicsNet: a web-based tool for creation and visual analysis of biological networks in 3D space". In: *Nucleic acids research* 46.W1 (2018), W514–W522.

[246] Johannes Zimmermann, Christoph Kaleta, and Silvio Waschina. "gapseq: Informed prediction of bacterial metabolic pathways and reconstruction of accurate metabolic models". In: *Genome biology* 22.1 (2021), pp. 1–35.

[247] Francisco Zorrilla et al. "metaGEM: reconstruction of genome scale metabolic models directly from metagenomes". In: *Nucleic Acids Research* 49.21 (Oct. 2021), e126–e126. ISSN: 0305-1048. DOI: 10.1093/nar/gkab815. eprint: https://academic.oup.com/nar/article-pdf/49/21/e126/41503923/gkab815.pdf. URL: https://doi.org/10.1093/nar/gkab815.

[248] Simon Portegies Zwart. "The ecological impact of high-performance computing in astrophysics". In: *Nature Astronomy* 4.9 (2020), pp. 819–822.