auto-p2docking: a pipeline for protein-protein docking

Mariana Cerqueira Barros,
Master in Bioinformatics & Computational Biology
Computer Science Department
Faculty of Sciences, University of Porto
2024

auto-p2docking: a pipeline for protein-protein docking

Mariana Cerqueira Barros

Dissertação realizada no âmbito do Mestrado em Bioinformática e Biologia Computacional

Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
2024

Orientador
Cristina Alexandra Gonçalves Paula Vieira, Investigador Auxiliar, Instituto de Investigação e Inovação em Saúde (i3S)
cgvieira@i3s.up.pt
Co-orientadores
Jorge Manuel de Sousa Basto Vieira, Investigador Principal, Instituto de Investigação e Inovação em Saúde (i3S)
jbvieira@i3s.up.pt
Alba Nogueira-Rodriguez, Doutorado, Instituto de Investigação e Inovação em Saúde (i3S); Departamento Ciência Computacional (CINBIO), Universidade de Vigo

Agradecimentos

Aos meus orientadores, Jorge e Cristina, pela paciência, dedicação e sabedoria com que me ensinaram e orientaram o meu trabalho durante este tempo. Por todas as vezes que cheguei ao laboratório e me disseram "não paniques Mariana, vai correr tudo bem" e acalmaram o meu coração nervoso, o meu sincero obrigado.

Á minha orientadora Alba pela simpatia e ajuda. Obrigada por me mostrar que não existem barreiras linguísticas na partilha de conhecimentos.

A toda a minha família, por me apoiarem, ajudarem a chegar até aqui e me fazerem rir com as parvoíces de sempre. Espero que estejam sempre presentes e orgulhosos de mim, nesta e nas minhas próximas conquistas.

Ao Pedro, por todo o teu amor, pela paciência necessária para me aturar, por todos os "vai correr tudo bem" e por nunca duvidares de mim, muito obrigada.

Finalmente, ao Matias, o meu bichinho, por todos os abraços e ataques surpresa e por todas as vezes que chegaste à minha beira a dizer PPIs para me fazer rir.

Resumo

O pipeline auto-p2docking é uma abordagem modular inovadora para a previsão da interação proteína-proteína (PPI) que ultrapassa as limitações da execução manual de múltiplos métodos *in silico*. Tira partido das diferentes bases de dados disponíveis no EvoPPI3 (bases de dados principais, predict interactors, polyQ_22 e Modifiers_22) para selecionar uma lista de interativos para uma determinada proteína. O Human Protein Atlas é então utilizado para verificar a expressão dos genes que codificam estas proteínas nos tecidos de interesse para o utilizador. Para cada proteína selecionada, o utilizador pode optar por utilizar métodos baseados na estrutura e na sequência para prever com precisão os resíduos de interação em soluções de acoplamento de proteínas. Containerizado em Docker, o Auto-p2docking garante a reprodutibilidade e reduz os erros através de uma configuração simplificada e da portabilidade do sistema. A sua arquitetura é composta por 22 módulos individuais que permitem: recuperar interactomas, obter estruturas proteicas 3D de diferentes bases de dados, determinar as semelhanças destas estruturas proteicas, aplicar diferentes metodologias de acoplamento e, se necessário, identificar os resíduos de interface previstos para cada proteína, selecionar as melhores soluções de acoplamento e explorar interfaces macromoleculares, identificar regiões de interação de acordo com as especificações do utilizador e visualizá-las no complexo proteico. Estes módulos podem ser especificados de acordo com as necessidades do utilizador, criando inúmeras possibilidades de pipeline. As principais características incluem o armazenamento de ficheiros intermédios para rastreabilidade, verificação automática de erros, um manual do utilizador e visualização gráfica do pipeline. Ao automatizar processos manuais e propensos a erros, o auto-p2docking reduz significativamente o tempo de análise e melhora a precisão. Para além de acelerar a análise de PPI, este pipeline promove a ciência aberta e a colaboração, disponibilizando ferramentas avançadas de bioinformática a um público mais vasto e fazendo avançar a investigação biomédica.

Palavras-chave: interações proteína-proteína (PPI), EvoPPI3, AlphaFold, D-I-TASSER, base de dados PDB, HADDOCK, HDOCK, FTDOCK, Pipeline

Abstract

The auto-p2docking pipeline is an innovative modular approach to protein-protein interaction (PPI) prediction that overcomes the limitations of manually performing multiple *in silico* methods. It takes advantage of the different databases available in EvoPPI3 (main databases, predict interactors, polyQ_22 and Modifiers_22) to select a list of interactors for a given protein. The Human Protein Atlas is then used to check the expression of the genes encoding these proteins in the tissues of interest to the user. For each protein selected, the user can choose to use structure- and sequence-based methods to accurately predict interacting residues in protein docking solutions. Containerized in Docker, auto-p2docking ensures reproducibility and reduces errors through simplified configuration and system portability. Its architecture consists of 22 individual modules that allow to: retrieve interactomes, obtain 3D protein structures from different databases, determine similarities of these protein structures, apply different docking methodologies and, if necessary, identify the predicted interface residues for each protein, select the best docking solutions and explore macromolecular interfaces, identify interacting regions according to user specifications and visualize them in the protein complex. These modules can be specified according to user needs, creating numerous pipeline possibilities. Key features include intermediate file storage for traceability, automated error checking, a user manual and graphical pipeline visualization. By automating manual and error-prone processes, auto-p2docking significantly reduces analysis time and improves accuracy. In addition to accelerating PPI analysis, this pipeline promotes open science and collaboration, making advanced bioinformatics tools available to a wider audience and advancing biomedical research.

Keywords: protein-protein interactions (PPI), EvoPPI3, AlphaFold, D-I-TASSER, PDB database, HADDOCK, HDOCK, FTDOCK, Pipeline

Índice

Lista de Figuras

Lista de Abreviaturas

AF - AlphaFold
AF2 - AlphaFold2
AR - Androgen Receptor
ATN1 - Atrophin-1
ATX1 - Ataxin-1
ATX3 - Ataxin-3
AutoDock - Automated Docking
CACNA1A - Calcium Voltage-Gated Channel Subunit Alpha1 A
CASP - Critical Assessment of Structure Prediction
CNS - Crystallographic and NMR System
C-scores - Confidence Scores
CNN - Convolutional Neural Networks
CRF - Conditional Random Fields
D-I-TASSER - Deep Learning-Based Iterative Threading ASSEmbly Refinement
DOT - Daughter of TURNIP
FFT - Fast Fourier Transform
FRODOCK - Fast Rotational DOCKing
FTDock - Fourier Transform Dock
GA - Genetic Algorithm
GOLD - Genetic Optimisation for Ligand Docking
HADDOCK - High Ambiguity Driven Biomolecular Docking
HDOCK - Hybrid DOCKing
HPmod - Human Proteome Model
HPA - Human Protein Atlas
HSYMDOCK - Hierarchical Symmetric Docking Algorithm
HTT - Huntingtin
I-TASSER - Iterative Threading ASSEmbly Refinement
ICM-DOCK - Internal Coordinate Mechanics Software
IPA - Invariant Point Attention
MDockPP - Hierarchical Protein Docking Algorithm
MC - Monte Carlo
MS - Mass Spectrometry
MSAs - Multiple Sequence Alignments
NNs - Neural Networks
NMR - Nuclear Magnetic Resonance
PDBe - Protein Data Bank in Europe
PDB - Protein Data Bank
PISA - Protein, Interfaces, Structures and Assemblies
polyQ - Polyglutamine
PPIs - Protein-Protein Interactions
PSIVER - Protein-Protein Interaction Sites Prediction Server
RFs - Random Forests
RMSD - Root Mean Square Deviation
RSA - Relative Solvent Accessibility
SCRIBER - Solvent Accessibility-Based Composite Residue Interaction Predictor Based on Evolutionary and Structural Features
SPPIDER - Solvent Accessibility-Based Protein-Protein Interface Identification and Recognition
SVM - Support Vector Machines
TBP - TATA-Box Binding Protein
Y2H - Yeast Two-Hybrid

1. Introduction

Proteins are complex molecules made up of one or more chains of amino acids. The sequence of these amino acids and their unique 3D structure determine their cellular function, making them essential building blocks in biology [1,2]. Protein-protein interactions (PPIs) occur when two or more proteins make specific contacts within a cell. These interactions require residues on one protein to make contact with residues on another protein, forming the interface of the PPI [3–5]. For interactions to occur, the proteins involved must not only have shapes that fit well together, ensuring high specificity of PPIs, but their amino acids must also have a certain degree of chemical complementarity, facilitating interactions such as hydrophobic, polar or charged interactions [4,6–8]. PPIs are crucial for maintaining the balance of the organism, allowing the formation of large molecular structures and enzymatic complexes. These interactions are fundamental to almost all cellular functions, including transport, signaling, catalysis of metabolic reactions, substrate binding, inhibition and biomolecular synthesis [3,9]. Given the importance of these regulatory processes for the proper functioning of the organism, any alterations affecting PPI integrity are of great interest for enriching the scientific understanding of biological processes.

1.1. PPI prediction and detection methods

PPIs are essential for the formation of large molecular structures and enzymatic complexes that play key roles in almost every cellular process [10]. When PPIs are disrupted, cellular dysfunction occurs, often leading to various diseases. PPIs have attracted attention as potential drug targets because of their high specificity in targeting disease-related pathways [10–13]. Understanding the entire set of PPIs in a cell or organism, known as the interactome, is critical to linking genotypes to phenotypes and, ultimately, understand how abnormalities lead to disease. Knowledge of specific sites on proteins where interactions occur, known as PPI sites, can reveal molecular recognition mechanisms and offer therapeutic opportunities to design drugs that regulate or mimic these interactions.

Experimental methods for identifying PPIs fall into two categories: large-scale screening methods and methods for studying individual interactions [10,14]. High-throughput methods, such as the yeast two-hybrid (Y2H) system, mass spectrometry (MS), protein chip and phage display, can screen many PPIs simultaneously by expressing each protein and exhaustively probing interactions, while methods such as X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy focus on specific PPIs and can determine PPI sites at the atomic level. The Y2H system exploits the modular nature of transcription factors to detect PPIs [15–18]. A “bait” protein is fused to a DNA binding domain and a “prey” protein is fused to an activation domain. The interaction between the bait and prey proteins reconstitutes a functional transcription factor that drives reporter gene expression in yeast cells. The Y2H system is advantageous for its high throughput, simplicity and low cost, but has a high false positive rate and may miss interactions that require specific post-translational modifications. MS identifies proteins that co-purify with a tagged bait protein by affinity purification followed by peptide analysis [16–19]. MS is highly sensitive, can analyze complex mixtures and provides quantitative data. However, it requires sophisticated equipment and expertise and may miss low abundance or transient interactions. Protein chips or microarrays immobilize many proteins on a solid surface and probe them with labelled proteins to detect interactions [16–18,20]. This method is quantitative and high throughput, analyzing thousands of PPIs in parallel. However, protein immobilization can affect conformation and function, and the technique requires specialized equipment and is expensive. In phage display, peptide or protein libraries are expressed on bacteriophage surfaces and screened for binding to a target protein [16–18,21]. High-affinity binders are selected and

amplified by multiple rounds of bio panning. Phage display is versatile and can identify high affinity binders but can miss interactions that require post-translational modifications and can be influenced by the phage surface, which affects protein folding. X-ray crystallography and nuclear NMR spectroscopy are used for detailed analysis. X-ray crystallography involves crystallizing the protein complex and analyzing the diffraction pattern to reconstruct a three-dimensional model at atomic resolution, providing detailed information about the interaction interfaces [16,22,23]. However, it requires high quality crystals and may not capture dynamic aspects of the interactions. NMR spectroscopy measures the magnetic properties of atomic nuclei to provide structural and dynamic information about proteins in solution [16,23,24]. It is useful for studying proteins in solution, providing insight into dynamics and flexibility, but is limited to smaller proteins or complexes, requires large amounts of protein, and can be time consuming and complex.

Experimental approaches, however, show limitations due to factors such as transient dynamics, post-translational modifications, proteins with disordered regions and physiological conditions [10,25–30]. In addition, proteins expressed in different cellular locations may not interact in vivo, even though they are able to interact in vitro. Experimental methods are also expensive, labor-intensive and time-consuming, and they have several limitations: 1) the resulting datasets are often noisy and prone to false positives [3,31]; 2) these methods can lead to failures in predicting certain types of protein interactions since they rely heavily on well-established experimental protocols in target organisms [3,32,33]; 3) since many transient and weak interactions are excluded due to the instability of these methods, this can result in inaccurate predictions [3,34]; 4) Large-scale methods often show significant discrepancies with each other, partly due to false positives and negatives [3,35–38].

Therefore, there is a need for *in silico* approaches to efficiently identify PPIs and PPI sites, expand coverage, and filter out false positives based on confidence scores of protein interactions [10]. Using computational methods to predict PPIs is not only less expensive compared to most large scale detection assays, but it also provides higher quality and confidence data sets and is significantly less labor intensive [3].

1.2. *In silico* approaches to identify PPIs

Computational predictors, also known as *in-silico* methods, use biological data to assess the likelihood of protein interactions [39]. They analyze different types of information, including protein sequences, structures, domain interactions, gene neighborhood, phylogenetic profiles and gene expression [32,39]. These methods make assumptions about interacting proteins. Sequence-based methods, can identify conserved motifs or domains in protein sequences that are known to mediate interactions (motif and domain analysis**)**, or co-evolving residues in proteins that suggest a physical interaction, based on the hypothesis that interacting proteins evolve together (co-evolutionary analysis), or learning algorithms to classify pairs of proteins as interacting or non-interacting based on features derived from their sequences, such as amino acid composition, physicochemical properties, and evolutionary profiles [40]. The recent sequence-based methods for predicting PPI sites uses deep learning methods such as convolutional neural networks (CNN) with a residue binding propensity to address data imbalance [41], and the DEep Learning Prediction of Highly probable protein Interaction sites (DELPHI) method which comprises an ensemble structure as a combination of CNN and recurrent neural networks [42]. Structural-based methods predict interactions by simulating the docking process between protein structures and scoring the binding affinities (docking simulations [43]), identifying potential interaction sites on protein surfaces using structural information (interface prediction [44]), or by inferring interactions based on the structural similarity of query proteins to known PPI complexes (comparative modeling [45]). Genomics and transcriptomics-based methods infer interactions based on

the co-expression patterns of genes across different conditions or tissues (gene co-expression analysis [46]), or by the identification of pairs of genes whose simultaneous knockout is lethal, suggesting a functional interaction (synthetic lethality analysis [47]). In network-based methods, new interactions are predicted from the protein network properties (e.g., degree, centrality; graph theory), or by using densely connected subgraphs (communities) within PPI networks that might represent functional modules (cluster analysis), or by combining PPI data with other biological networks (e.g., gene regulatory networks) to improve prediction accuracy (data integration). Literature based models, can also be used to infer novel PPI based on proteomic data available in curated database aggregators such as EvoPPI [48,49]. The integration of various computational approaches, could in principle, improve prediction accuracy and reliability, to obtain a comprehensive understanding of protein interactions and their roles in cellular functions.

Among the most reliable but also more time-consuming methods are the structure-based methods that uses molecular docking, different empirical scoring functions, physics-based methods, knowledge-based approaches, or quantitative structure–activity relationship methods to model structural orientations of the two interacting proteins, to determine their binding affinity. It uses the structures of individual proteins to infer their binding orientations, and then binding free energy between the interacting proteins are estimated [50,51]. In the modelling search the target protein (receptor) is stationary and the ligand moves around it, in all possible orientations, by using a fast Fourier transform (FFT) approximation (as implemented in ZDOCK [52]; GRAMM [53]; Daughter of TURNIP (DOT) [54] ; SmoothDock [55]; ClusPro [56,57]; MolFit [58]; Fourier Transform Dock (FTDock) [59]; 3D-Dock [59–61]; PIPER[62]; pyDock [63]; Hybrid DOCKing (HDOCK) [64,65]; SDOCK; HEX [66]; Fast Rotational DOCKing (FRODOCK) [67]; InterEvDock [68]; hierarchical protein docking algorithm (MDockPP) [69]; CoDockPP [70]; hierarchical symmetric docking algorithm (HSYMDOCK) [71]; and SAM [72]), genetic algorithm (GA; such as Automated Docking (AutoDock) [73] , DARWIN [74], Multi-LZerD [75], and Genetic Optimisation for Ligand Docking (GOLD) [76]), or Monte Carlo (MC; such as RosettaDock [77], Internal Coordinate Mechanics software (ICM-DOCK) [78], and High Ambiguity Driven biomolecular DOCKing (HADDOCK) [79]) [80]. To restrict the sampling search, predictions for interacting residues can be used for each protein as unbound monomer surfaces. For that, structure-based methods can be used, where features such as solvent accessibility, surface curvature, electrostatic potentials and secondary structure, together with machine learning algorithms such as support vector machines (SVMs), neural networks (NNs), random forests (RFs) and probabilistic graphical models are used (ISPRED4 and SPPIDER (http://sppider.cchmc.org/) [39,81]). Sequence-based methods, that use evolutionary information, conservation scores, physicochemical properties and sometimes structural features calculated from sequences can also be used, such as SCRIBER (http://biomine.cs.vcu.edu/servers/SCRIBER/) and PSIVER (https://mizuguchilab.org/PSIVER/) [82,83] .These methods, are available at C-PORT [84]and can be used in combination. Methods based on chemical thermodynamics (that depend on parameters such as the interface area, residue/atom composition and contacts, hydropathy index charge distribution, topological complementary, among others), such as the Protein, Interfaces, Structures and Assemblies (PISAePDB [85]) are then used to identify biological relevant contacts, from the other crystal contacts present in the PDB file of the protein complex. Understanding the molecular details of protein interactions, is crucial for unravelling cellular processes, designing drugs, and understanding disease mechanisms.

For Structure based analyses, 3D structures are needed, but such structures have been determined for a fraction of all proteins only (using mostly X-ray crystallography and NMR spectroscopy, available at the Protein Data Bank (PDB) database). Therefore, Structure-based molecular docking often uses predicted structures of individual proteins. Predictions can be obtained using sequence homology (Rosetta [86], I-Tasser (Iterative Threading ASSEmbly Refinement) [87] and FragFold [88])

co-evolutionary information [89], neuronal network modeling and deep machine learning (DeepMind's artificial intelligence mode as in AlphaFold [90] (AF)) have been used to predict with very high accuracy structure predictions (see for a review [91]). AlphaFold2 (AF2) combines alignment-based predictors together with protein language model-based predictors and deep learning models for sequence-based protein structure predictions [92]. This prediction methodology has been used to predict the 3D structure of the majority of the UniProt sequences, available at AF Protein Structure Database [90]. Although AF models have remarkable ability to predict protein architecture, they can present small side-chain variations, that impede docking within important binding sites, thus raising doubts if AF models can be used for molecular docking purposes [93]. Other methodologies such as Deep learning-based Iterative Threading ASSEmbly Refinement (D-I-TASSER) refine AF structures using complementary deep neural-network predictors, to improve enrichment factors. Data bases for human proteins using this methodology are also available at Human Proteome Model (HPmod; https://zhanggroup.org/HPmod/about.html). It should be noted that proteins have different structural conformational states, that can highlight specific binding sites in different biological states of the proteins, and that a single structural model thus not account for all variation. Therefore, the different 3D inferences can be relevant in different biological processes, and they should be complemented with experimental data.

1.3. Structural based methods for the prediction of PPIs

Prediction of PPIs is fundamental to the understanding of biological processes and cellular pathways. These interactions play an essential role in various cellular functions such as signal transduction, gene regulation, immune response and metabolism. The prediction of PPIs involves several methods and tools, starting with 1) the search for the interactors of the ligand protein using primary protein databases, such as those available in EvoPPI. Further support for the retrieved PPI can be provided by inference from other model species, if conserved interactions between pairs of proteins that have interacting homologs in another organism are conserved between distantly related species. In the case of neurodegenerative diseases caused by abnormal expansion of the polyglutamine (polyQ) tract, other databases such as the PolyQ_22 (from patients, human cell lines used to study these diseases, model species mutants expressing a human protein of interest), and modifiers_22 (reporting protein data from cross-species genetic screens) are available in EvoPPI3 [48,49] that can be used to support the PPI. In addition, expression data of the genes encoding the interactors in tissues of interest can be used to support the PPI. The following tools can be used for these tasks.

1) Search for data on the target protein: before starting any analysis, it is crucial to obtain information about the protein of interest, such as its three-dimensional structure and any data on known interactions.

2) Predict the three-dimensional structure of the protein: predicting the three-dimensional structure of the protein is a fundamental step as it provides detailed information about the spatial conformation of the protein. Tools such as PDB[94], AF[90] and D-I-TASSER[95] are used for this purpose.

3) Structural alignment: structural alignment is performed to compare the three-dimensional structure of the protein with other proteins to identify similar or conserved regions and to ensure that the 3D structures we will use in our analysis are appropriate. TM-Align [96] can be used to perform this alignment.

4) PPI interface residue prediction: identification of PPI interface residues is critical for predicting the regions where proteins interact. Tools such as CPORT[84] are used for this prediction.

5) Protein docking prediction: protein docking is used to predict how proteins interact with each other by determining their relative orientations and conformations in

the protein complex. Tools such as HDOCK[65], PyDock[63] and HADDOCK[79] are commonly used to predict these interactions.

6) Protein structure analysis: once the protein-protein complex has been predicted, it is essential to analyze the resulting structure to better understand the characteristics of the interactions. Tools such as PDBePISA[85] and CCP4[97] can be used for this analysis.

1.3.1. Search for available data on the target protein

Before studying a protein's interactions, it is crucial to perform a data search to understand its structure, function, and biological relevance. This involves identifying its amino acid sequence, determining its three-dimensional structure, studying its expression and cellular localization, and exploring its interaction with other molecules. This preliminary investigation provides a solid basis for studying protein interactions and helps to contextualize their role in biological processes.

1.3.1.1. EvoPPI3

PPIs play crucial roles in coordinating molecular functions both within individual cells and across biological systems. Consequently, they have been extensively investigated using diverse methodologies [98,99]. The outcomes of these studies have been systematically cataloged into repositories known as PPI primary databases, such as BioGRID [100,101], CCSB [102], DroID [103], FlyBase [104], HIPPIE [105], HitPredict [106], HomoMINT [107], Instruct [108], Interactome3D [109], Mentha [110], MINT [111,112], and PINA [113]. In EvoPPI3, an open source web application tool, that allows the user to easily select and retrieve the available PPI data for a given protein, these databases are assigned as Databases, and the user can select all or a subset of them. EvoPPI3 Databases are available for 10 species (*Homo sapiens*, *Mus musculus*, *Rattus norvegicus*, *Bos taurus*, *Oryctolagus cuniculus*, *Gallus gallus*, *Xenopus laevis*, *Danio rerio*, *Drosophila melanogaster*, and *Caenorhabditis elegans*), and GeneID of the gene that encodes the protein of interest is used to perform the search. Since interacting protein pairs tend to be evolutionarily conserved when comparing different species [114], the PPI information obtained for one species can in principle be transferred to another species if orthologous gene pairs can be identified, for example using a BLAST-based approach [48,98]. In EvoPPI3, the user can also obtain the predicted interactions of a protein from a different species [98,99]. In this case, data is available for *H. sapiens*, *M. musculus*, *D. rerio*, *D. melanogaster*, and *C. elegans*.

In EvoPPI3, there are also other databases. called polyQ_22 and modifiers_22, concerning proteomic data of the nine proteins (huntingtin (HTT), androgen receptor (AR), atrophin-1 (ATN1), ataxin-1 (ATX1), ataxin-2, ataxin-3 (ATX3), calcium voltage-gated channel subunit alpha1 A (CACNA1A), ataxin-7 and TATA-box binding protein (TBP)) that cause polyQ diseases when they have an abnormal expansion of the CAG trinucleotide repeat. The polyQ databases combine proteomic data from patients, model species mutants expressing a human protein of interest, as well as human cell lines used to study polyQ diseases. On the other hand, modifiers_22 databases report proteomic data from gene modifier experiments [48]. EvoPPI3 represents a major advance in the study of polyQ diseases by integrating polyQ interactome and predictome datasets, which are critical for understanding disease mechanisms, identifying potential therapeutic targets and accelerating research efforts towards effective treatments for these neurodegenerative disorders.

1.3.1.2. The human Protein atlas

Proteins are the essential building blocks of life, and knowing the spatial distribution of all human proteins at the organ, tissue, cellular and subcellular levels would greatly enhance our understanding of human biology in health and disease [115]. Using antibody-based proteomics and integrating several other omics technologies, the Human Protein Atlas (HPA) project is a large-scale initiative to map the entire human proteome [116].

The HPA serves as a comprehensive 'atlas' of human proteins, aiming to provide a detailed overview of the spatial distribution and expression of every human protein across different human tissues, cancer types and cell lines. It achieves this through the integration of multiple omics technologies, including antibody-based imaging, MS, proteomics, transcriptomics and systems biology [115,116]. Version 23.0 of the HPA is organized into twelve distinct sections, each focusing on different aspects of analyzing human proteins on a genome-wide scale: 1) Tissue Section, 2) Brain Section, 3) Single Cell Type Section, 4) Tissue Cell Type Section, 5) Pathology Section, 6) Disease Blood Atlas Section, 7) Immune Cell Section, 8) Blood Protein Section, 9) Subcellular Section, 10) Cell Line Section, 11) Structure Section and 12) Interaction Section.

The HPA is a vital resource for understanding how proteins contribute to the formation of organelles, cells, tissues, and organs. It makes a significant contribution to both basic and clinical research [115].

1.3.2. Three-dimensional protein structure prediction

Inside every cell in our bodies are billions of tiny molecular machines called proteins. Proteins are the building blocks of life, and understanding their structure can facilitate a mechanistic understanding of their function which is crucial for drug development, disease understanding and enzyme design [90,117–119]. The problem of protein folding, which refers to how proteins adopt their three-dimensional shapes, is a major scientific challenge. The PDB serves as a repository of experimentally determined protein structures, providing vital information for researchers worldwide [94]. Despite their precision, experimental techniques such as X-ray crystallography and NMR spectroscopy are costly and time-consuming, resulting in a significant gap between the number of known protein sequences and experimentally determined structures [120]. To bridge this gap, scientists are increasingly turning to computational approaches to protein structure prediction, which hold great promise for advancing our understanding of proteins and their functions. The fast progress and effective implementation of deep learning in protein structure prediction has led to new methods in this field[120]. The CASP (Critical Assessment of Structure Prediction) competition, held biennially, plays a crucial role in advancing protein structure prediction, since evaluates predictions against experimentally solved structures, maintaining double-blinded procedures to ensure fairness and impartiality [121,122].

I-TASSER is a computational tool for predicting the three-dimensional (3D) structure of proteins from their amino acid sequences [87]. It uses a combination of threading, structural assembly and iterative refinement methods to produce high quality protein structure predictions [87,123,124]. The process begins by threading the target protein sequence through a database of known protein structures to identify template proteins with similar sequences. This involves aligning the target sequence to several template structures to find regions of local similarity. Once suitable templates have been identified, the next step is to assemble these fragments into a full-length 3D model using MC simulations that optimize the spatial arrangement of secondary structure elements. After constructing an initial model, I-TASSER refines the structure through iterative simulations that include energy minimization and the use of knowledge-based potentials to improve the accuracy of the model. I-TASSER provides confidence scores (C-scores)

for the predicted models, indicating their reliability, with higher C-scores corresponding to models with higher confidence. In addition to structure prediction, I-TASSER also predicts the biological function of the protein by comparing the predicted structure with known structures in the protein function database. I-TASSER is widely recognized for its accuracy and reliability in predicting protein structures, often placing highly in the CASP competitions [87,91,125].

AF (https://alphafold.ebi.ac.uk) is an end-to-end protein structure prediction method that use deep learning techniques to predict the 3D structure directly from the amino acid sequence, revealing the intrinsic connection between the input sequences and the resulting structures [120]. AF, developed by DeepMind, has revolutionized protein structure prediction since its launch in 2020 [90]. It accurately predicts protein structures from amino acid sequences. AF DB extends this capability by providing predictions for over 200 million proteins in UniProt. AF2, the latest iteration presented at CASP14, has set the standard for structure prediction accuracy [120,122,126]. Using state-of-the-art deep learning algorithms and evolutionary conservation principles, AF2 uses an end-to-end deep neural network trained to generate protein structures directly from amino acid sequences, incorporating information from homologous proteins and multiple sequence alignments (MSAs). In the CASP14 evaluation, AF structures significantly outperformed the competition, demonstrating remarkable accuracy [120,122]. Key components contributing to AF2's success include the Evoformer, an attention-based network that iteratively refines raw MSAs and structure templates, and the Invariant Point Attention (IPA) module, that connects directly to the Evoformer and facilitates reasoning about spatial (3D structure) and evolutionary (sequence) relationships. Recycling iterations for structure refinement (where the output from one iteration is fed back into the network for further refinement), along with additional factors such as self-distillation (the model uses its own predictions as part of the training data to improve its performance), enriched MSA from metagenome databases, MSA clustering and extensive training with tensor processing units (specialized hardware accelerators designed to speed up machine learning tasks) facilities, further enhance AF2's performance [117,120,122,126]. However, AF2 has limitations, particularly in the prediction of orphan proteins, where confidence scores rely heavily on the presence of homologues in the PDB. While AF2 dominated CASP14 with unparalleled prediction accuracy, its supremacy was challenged in CASP15, where over 40 teams outperformed it in terms of accuracy [120,122]. Interestingly, many of the top teams in CASP15 incorporated AF2 into their prediction methods to varying degrees, highlighting the profound influence of AF2 in the field of protein structure prediction. Despite this shift in performance dynamics, AF2 continues to stand out as one of the most advanced methods in the field.

Despite the success of end-to-end methods, the two-step approach remains valuable as it requires fewer computational resources and remains accessible to labs with limited hardware [120]. This approach uses deep learning methods to infer spatial relationships within protein structures, including contacts, distances, orientations and hydrogen bonds between amino acids [120]. These inferred relationships are then combined with either knowledge-based or physics-based force fields. Optimization techniques, such as MC-based methods, are then used to explore the most energetically favorable conformations [95,123,127]. D-I-TASSER, an advanced protein structure and function prediction tool developed by the Yang Zhang laboratory, extends the capabilities of the I-TASSER method. D-I-TASSER integrates a set of neural network predictors to predict spatial relationships between amino acids such as contacts, distances and hydrogen bonding patterns [120]. These predictions are incorporated into the I-TASSER force field and guide the assembly of fragments to produce complete protein structures. The function prediction aspect is based on COFACTOR [128], which uses hybrid models that combine information from structural and sequence similarity as well as PPIs networks for optimal function prediction. At CASP15, the D-I-TASSER pipeline, called "UM-TBM", achieved first place in the interdomain prediction category by using different

strategies for MSA generation, integration of AF2 and templates identified by LOMETS3 [120,129]. LOMETS3 (https://zhanglab.ccmb.med.umich.edu/LOMETS/), represents an innovative meta-server approach to template-based protein structure prediction and function annotation. It integrates newly developed deep learning threading methods to improve prediction accuracy and functional understanding. D-I-TASSER marks a significant leap forward in protein structure and function prediction, demonstrating the synergy between deep learning techniques and traditional computational methods [120]. Its outstanding performance at CASP15 demonstrates its effectiveness in dealing with the complexity of interdomain prediction, consolidating its position as a leading solution in structural bioinformatics [91].

1.3.3. Structural Alignment

Insights into structural similarities are crucial, especially when proteins share an evolutionary history. Understanding these relationships can drive advances in structural biology, aiding protein fold classification, protein structure modelling, and structure-based protein function annotation [3,130–133]. Protein structure alignment methods typically consist of two main components: a scoring function that measures similarity, and a search algorithm to optimize the scoring function. Over the past two decades, numerous computer programs have been developed for pairwise and multiple structure alignment. Structure comparison provides insights into the distant past of protein evolution that sequence comparison alone cannot provide. Different methods use different representations, scoring functions and optimization algorithms, often leading to conflicting results even for moderately distant structures. Sequence similarity could be a good predictor of function because it determines the 3D structure, but this isn't always the case since different amino acid sequences can lead to very different structures, and similar sequences can lead to dissimilar structures. Therefore, structural similarity is a more reliable predictor of functional similarity than sequence similarity. Structural similarity between proteins often indicates functional similarity, making it a reliable predictor of function. Structural alignments typically focus on the common core, distinguishing it from regions where one-to-one structural correspondence cannot be established [134]. Structural alignment methods include rigid, flexible, and elastic aligners, each of which handles structural variations differently. Structural mutations, insertions and deletions are accommodated by deformations of the common core, preserving the precise geometry of the active site, while peripheral regions may undergo complete refolding [134]. Therefore, structure comparison methods that allow for flexibility and plasticity generate the most biologically relevant alignments. For a match to occur between two structures needs to have a correspondence between elements of the structures and a rigid body transformation where the Root Mean Square Deviation (RMSD) between elements is less than some threshold ε. Once a structural similarity score is chosen, the alignment problem is to find the optimal set of correspondences.

Computational assessment of structural similarity is challenging due to the ill-posed nature of measuring partial similarity. There are many ways in which two structures can be similar, and depending on the application, certain aspects may be more important than others. Defining a quantitative measure of structural similarity is difficult because there's no consensus on which aspects of the structure are most important. Specific algorithms are suitable for different applications and there isn't a single best solution. TM-align [96] is an algorithm for sequence independent protein structure comparisons which relies on residue-to-residue alignment. It obtains an optimal superposition of the two aligned structures and returns a TM-score [135]. This score ranges from [0,1], 1 being a perfect match between structures. This scale, contrarily to RMSD, is insensitive to the local modelling error. A score of >0.5 indicates a model of

correct topology while <0.17 means a random similarity [96]. TM-align is one of the most used computational methods for structure alignment.

1.3.4. Prediction of protein interface residues

Based on the strength and stability of their interactions, protein-protein complexes can be divided into two main categories: obligate and transient assemblies [136,137]. Obligatory complexes are functional only when the monomers (individual protein units) are bound together, and these monomers do not exist as stable entities on their own in vivo. Obligatory complexes have a high degree of shape complementarity, with interface residues that are like the hydrophobic core found in globular proteins (similar to the core regions of proteins that are buried away from water). In contrast, transient complexes are formed by proteins that are functional even when they are not bound. These complexes are stabilized by weaker interactions, have less geometric complementarity (the interacting surfaces do not fit together as precisely as in obligatory complexes), and the interface area is generally smaller than in obligatory complexes. In addition, the hydrophobicity of residues at the interface of transient complexes is like that of the rest of the protein surface. Interfacial sites in obligatory complexes are generally easier to detect due to their larger size, flatter shape, higher hydrophobicity and greater conservation compared to those in transient interfaces.

Proteins interact through interface sites, which consist mainly of surface residues (amino acids on the surface of the protein that are accessible to the surrounding environment) [136,138]. These residues tend to be more conserved than others because they are important for protein function, but this conservation signal weakens below a certain level of solvent accessibility because residues with high solvent accessibility are more exposed to the environment. Therefore, the accurate definition of surface residues is crucial for building databases that exploit evolutionary conservation. The accuracy of predictions is highly dependent on the criteria used to classify surface residues. Typically, residues are classified as surface residues if their relative solvent accessibility (RSA) is above a certain threshold. Different studies use different cut-offs, usually ranging from 5 to 16%, with higher thresholds resulting in fewer residues being classified as solvent exposed [136]. By comparing regions where proteins can interact (interfacial regions) with regions where they do not interact, researchers can identify specific features that are common to interfacial regions. These are inherent properties of the residues involved in forming PPIs that help predict where these interactions are likely to occur [136–138].

1) Structure-based features: these features are derived from the tertiary structures of target proteins and include solvent-accessible surface area, secondary structure, local geometries and the spatial distribution of hydrophobic and polar surface patches.

2) Sequence-based features: these features are derived from the amino acid sequence alone and use various physicochemical properties to identify interface regions, such as hydrophobicity and electrostatic desolvation, as well as structural attributes predicted from the sequence, such as secondary structure and solvent accessibility.

3) Evolutionary features: these are calculated by comparing the sequence of a query protein with that of its homologues. Interface residues are generally more conserved than non-interface surface residues, which are under less selective pressure. Sequence conservation reflects evolutionary selection at interface sites to maintain protein function. These features have high discriminatory power for identifying interface residues.

Protein interface prediction techniques can be broadly categorized into sequence-based and structure-based approaches. Structure-based methods rely on data from 3D protein structures or models of the proteins involved [137]. These methods

define a surface patch for each target residue in a protein structure, which can be defined in two ways: 1) Fixed radius: the surface patch includes the target residue and all surface residues within a certain radius and 2) Fixed number of neighbors: the surface patch consists of the target residue and its K nearest surface residues, where K is a predefined constant. Each surface patch is represented as a vector containing various structural and sequence-derived features. The classification of each target residue within the surface patch is either interfacial (1) or non-interfacial (0). An example of a structure-based predictor is Solvent accessibility-based Protein-Protein Interface iDEntification and Recognition (SPPIDER) [139], which uses the difference between the predicted and actual RSA of a residue as a feature to predict interfaces. SPPIDER combines the output of multiple neural networks using majority voting to improve prediction accuracy. ISPRED4 [140] is another structure-based method that combines SVMs and Conditional Random Fields (CRF) and integrates conservation and co-evolution features of each protein partner among other sequence and structure-based descriptors [141]. The CRF layer was used to capture possible correlations between neighboring positions, which are otherwise treated independently by the SVM step.

While structure-based predictors only need to identify interfacial residues among surface residues, they have certain limitations: 1) these methods require knowledge of the protein structures in question, which is often not available for proteins involved in transient interactions [137,142]; 2) Structural features extracted from unbound proteins may differ in bound complexes due to conformational changes induced by binding [137,138].

The prediction of protein interfaces from sequence information alone is very difficult, leading to the lack of development of sequence-based predictors [137]. For a protein sequence with L residues, a fixed width sliding window (usually 3-30 residues) is used to generate overlapping windows centered on the target residues. These windows are used as input feature vectors, often in the form of physicochemical, statistical or predicted structural features. Protein-protein interaction SItes prediction seVER (PSIVER) [83] is a representative sequence-based predictor that uses a position-specific scoring matrix (PSSM) and predicted accessibility as inputs to a Naive Bayes classifier. Solvent accessibility-based Composite Residue Interaction predictor Based on Evolutionary and Structural features (SCRIBER) [82] integrates a variety of biological data sources, utilizing solvent accessibility information, evolutionary conservation metrics and structural features of proteins. SCRIBER uses a Multi-Level Logistic Regression approach to effectively identify which residues on the surface of a protein are likely to be involved in interactions with other proteins.

Currently, structure-based machine learning predictors generally have higher accuracy than sequence-based methods due to several factors [137]: 1) Structure-based methods can easily identify surface residues and ignore internal residues. 2) Many protein-protein interfaces involve residues that are close in 3D space but far apart in sequence. 3) Information on spatial positions and geometric complementarity is readily available from 3D structures, enhancing the predictive power of structure-based methods. While structure-based methods for predicting protein interfaces offer higher accuracy and several advantages, their reliance on 3D structural data limits their applicability [137]. Sequence-based methods, although less accurate, are crucial for predicting interfaces in proteins lacking structural information.

1.3.5. Protein-protein Docking

PPIs regulate the intricate molecular relationships within organisms and consequently dictate biological function at cellular and systemic levels [143]. PPIs can be identified using high-throughput experimental techniques. However, these methods have limitations in terms of cost, time and false positives, resulting in a relatively small number of experimentally determined protein-protein complex structures compared to individual protein structures in the PDB [64,144]. To complement experimental methods, computational docking techniques have been developed. Structural based docking approaches start with 3D protein structures to predict the 3D protein-protein structure, and perform docking by exploring potential binding configurations of one protein relative to the other, with a level of accuracy similar to experimental methods [143,145]. These configurations are ranked based on binding scores determined using an energy scoring function. Global protein-protein docking, also known as ab initio or free docking, is required to explore potential binding configurations in all six degrees of freedom (three translational and three rotational) [64]. This approach was more common in the early stages of protein-protein docking development, when experimental complex structures were limited and there wasn't much binding site information available [64,146,147]. However, with the advancement of structural proteomics projects, more protein-protein complex structures are being determined experimentally, providing valuable information about binding interfaces. In addition, information about interacting residues between proteins can be inferred from sequence data as sequence databases continue to expand. Using this binding site information, a new approach called template-based docking has emerged that incorporates binding interface information.

The docking method is chosen according to the docking problem [148]. If X-ray structures are available for all proteins involved or their closest homologs, free docking methods can be used. However, with the increasing number of protein complex structures in the PDB, it's becoming possible to predict related protein complexes using template-based and homology modelling methods even without the structures of the component proteins. In principle, if a suitable template is available in the PDB, individual protein complex structures can be modelled directly using template-based docking alone [64]. However, the reliability of the binding interface in the chosen template is crucial, as homologous proteins may not always share similar binding interfaces when they form a complex. In such cases, template-based docking may lead to incorrect predictions. Template-based methods generally yield more accurate predictions when reliable templates are available [64,148]. Nevertheless, template-free docking remains valuable when appropriate templates are lacking. Unlike template-based docking, which relies on biological information, template-free docking globally explores all potential binding modes and can identify true binding modes. In essence, both template-based and free docking methods have their advantages and limitations in predicting complex structures. Docking approaches can also be categorized as flexible or rigid. In flexible docking, one protein remains fixed while the other rotates and translates around it; a score is calculated for each rotation and the most favorable pose is selected at the end [79]. Due to the large number of degrees of freedom in the backbone and side chains, proteins are often treated as rigid bodies to facilitate a global search [80,84]. In rigid docking, the internal geometry of the proteins remains fixed, preventing any spatial changes during the docking process; the ligand is explored in a six-dimensional rotational or translational space. While the number of degrees of freedom in this type of docking is lower, the large number of translations and rotations in six-dimensional space highlights the importance of an efficient search strategy [79,149–152].

1.3.5.1. pyDOCK

PyDock[63] is a free docking server with a rigid-body docking algorithm based on the use of FFT [59]. Its unique feature is the combination of electrostatic, desolvation and van der Waals terms, with weighting factors initially optimized for a small set of cases to avoid overfitting [63,153]. FTDock uses an algorithm that discretizes two molecules on orthogonal meshes and performs a global scan of translational and rotational space It assesses the fit between the surfaces of the two meshes, known as surface complementarity. To speed up these computations, FTDock uses FFT, which replace the convolutions of the grids with multiplications in Fourier space. The distributed version for local execution is designed to integrate seamlessly with the rigid body docking sets provided by FTDock 2.0 [59]. PyDock has consistently demonstrated strong predictive performance in community-wide evaluation experiments such as CAPRI or CASP, and has provided valuable biological insight and interpretation of experiments by modelling numerous biomolecular interactions of interest [153].

1.3.5.2. HDOCK

HDOCK [65] is a hybrid docking protocol that combines both template-based and template-free approaches with the aim of retaining the strengths of both docking methods while minimizing their weaknesses, improving docking accuracy [64]. Instead of treating proteins individually, it uses homologous complexes as templates. First, it identifies homologous complexes using a BLAST search with a 25% sequence identity cut-off. It then models the proteins using these complexes as templates, either by superimposing structures or using homology modelling. Next, a 1000-step MD simulation using AMBER [154] optimizes the modelled complexes to remove atomic clashes. The structures are separated and subjected to a hierarchical docking algorithm to explore binding modes. Symmetric docking is performed by constraining translations [64]. In the absence of templates, MODELLER [155] is used to model based on homologous monomers. Finally, binding modes are ranked by energy scores and clustered [64]. HDOCK combines template information with traditional docking methods to accurately predict PPIs.

1.3.5.3. HADDOCK

In recent years, numerous docking-based methods have been developed to study protein complexes. Many of these are based on energetics and shape complementarity rather than experimental data. HADDOCK [79], developed by BonvinLab, incorporates biochemical and/or biophysical interaction data, such as chemical shift perturbations from NMR titration experiments or mutagenesis data. These data are used as Ambiguous Interaction Restraints (AIRs) to guide the docking process [79,145]. An AIR is defined as an ambiguous distance between all residues that have been shown to be part of the interaction. The docking protocol consists of three stages: rigid body energy minimization, semi-flexible refinement in torsion angle space and final refinement in explicit solvent. At the end of each stage, the structures are scored and ranked, and the best structures are passed on to the next stage. During the torsion angle refinement, the amino acids at the interface (side chains and backbone) are allowed to move in order to optimize the packing of the interface. The HADDOCK score is a weighted sum of van der Waals, electrostatic, desolvation and restraint violation energies along with the buried surface area. Users can adjust various parameters that control the docking/refinement protocol, such as temperature, number of time steps and force constants, as well as the number of structures and scoring weights for each stage. HADDOCK uses CNS (Crystallographic and NMR System) as its structure calculation engine. HADDOCK has shown excellent performance in CAPRI experiments and is one of the most popular methods for protein docking [145,156,157].

1.3.6. Protein Structure analysis

Understanding how proteins function in cells, during infections and in disease is crucial to many areas of science and medicine [158]. Scientists gain valuable information about the function of proteins from their amino acid sequences, structures and the molecules with which they interact. With the advent of next-generation sequencing, we have access to millions of unique protein sequences from more than 8000 fully sequenced genomes. In addition, improvements in crystallography have led to a growing number of protein structures available in databases such as the PDB [144], which now totals over 190,000 structures. Compared to sequences alone, protein structures provide a deeper insight into biological function because they show how the protein's building blocks are arranged in three dimensions [158]. They also show which parts of the protein interact with other molecules. Although the availability of protein structures is increasing, it's still slower than the availability of protein sequences. Tools such as PDBePISA [85] and Collaborative Computational Project Number 4 (CCP4) [97,159] are widely used to analyze protein interactions and predict how proteins assemble into larger structures.

PDBePISA is a web server and database developed by the Protein Data Bank in Europe (PDBe) for the analysis of macromolecular complexes [85,158]. It provides information on PPIs, including the identification of interfaces, the calculation of interaction surfaces and the prediction of quaternary structure assemblies. PDBePISA uses crystallographic symmetry and surface area calculations to determine the biological assemblies of macromolecules and assess their stability. CCP4 is a suite of programs widely used in structural biology for the analysis, modelling and visualization of macromolecular crystallography data [158,159]. It provides a comprehensive set of tools for protein crystallography, including programs for data processing, phasing, model building, refinement and validation. CCP4 aims to facilitate the interpretation of X-ray diffraction data and the determination of macromolecular structures by providing an integrated software package with state-of-the-art algorithms and methods.

1.4. Advantages of pipelines

Pipelines are structured sequences of data processing steps designed to transform raw data into meaningful results [160,161]. Each step in a pipeline typically represents a specific task, such as data cleansing, transformation, analysis and visualization. By automating the flow of data through these tasks, pipelines ensure that the process is consistent and repeatable. They can be implemented using a variety of tools and programming languages, depending on the specific requirements of the analysis.

Reproducibility is a fundamental part of scientific research, allowing experiments and analyses to be repeated and validated by different researchers under the same conditions [160,161]. Pipelines play a critical role in achieving this by standardizing and automating the workflow. Consistency is a key benefit of pipelines. When a pipeline is executed, it ensures that each step of data processing and analysis is performed in a consistent manner every time. This is essential for achieving reliable and comparable results, as manual processes can introduce variability and errors. Automated pipelines reduce the likelihood of such discrepancies, thereby increasing the integrity of the results. Automation is another key benefit of using pipelines. Data analysis often involves repetitive tasks that are prone to human error when performed manually. Pipelines automate these tasks, which not only reduces the risk of error, but also saves significant time. This allows researchers to focus on interpreting results and making scientific inferences, rather than on the mundane aspects of data handling. Transparency in the research process is greatly enhanced by well-documented pipelines. Each step in a pipeline is clearly defined and recorded, providing a transparent record of the entire data analysis workflow. This documentation is essential for understanding how results were

obtained and for verifying the methodology used. It ensures that other researchers can follow the same process and achieve similar results, which is a fundamental aspect of scientific research. Another important feature of pipelines is their scalability. Modern scientific research often involves large amounts of data, such as those generated in genomics, neuroimaging and environmental studies. Pipelines are designed to handle such large amounts of data efficiently. They can be scaled up or down to handle datasets of varying size and complexity, making them highly adaptable to different research needs. Pipelines also facilitate collaboration between researchers. In collaborative projects, team members can work on different parts of a pipeline and their contributions can be easily integrated into a cohesive workflow. This standardized framework ensures that everyone is on the same page, which is critical to the success of collaborative research efforts. Finally, pipelines offer the benefit of version control. They can be versioned, allowing researchers to track changes over time and revert to previous versions if necessary. This is particularly useful when new data or methods are introduced, as it helps to maintain the integrity of the analysis. Version control ensures that the research process remains transparent and that any changes are well documented and justified.

Pipelines are essential tools for scientific analysis. They provide a structured, automated and transparent approach to data processing and analysis, enabling researchers to produce consistent, reliable and scalable results. By facilitating collaboration, ensuring reproducibility and maintaining detailed documentation, pipelines advance scientific knowledge and build confidence in research results. The structured nature of pipelines not only increases the efficiency and reliability of scientific research, but also promotes the transparency and reproducibility that are essential to the progress of science.

1.5. Containerization with Docker

Docker is an open-source platform that simplifies the process of building and deploying applications [162]. Released to the public in 2013, Docker has become the leading container technology, allowing applications and all their dependencies to be packaged into a standardized format called a container. These containers run in isolation on top of the operating system kernel [162,163]. Docker makes it easy for developers to create and manage containers, allowing applications to be packaged into lightweight containers that can run anywhere without modification[162]. When building a Docker image, a Docker file contains instructions that are executed when the "docker build" command is run. Docker images are stored in Docker registries, like source code repositories, where they can be pushed or pulled. There are public and private repositories; Docker Hub is a public repository where users can share and access images. Docker containers are created from Docker images and contain everything needed to run an application in isolation. The benefits of Docker containers include speed, portability, scalability, rapid deployment, and density [162,164]. Containers are quick to create and can be moved quickly through development, test, and production environments. They are highly portable and maintain performance regardless of the environment. Standardized container formats provide predictability and reliability in deployment, making it easier for administrators and developers to manage applications. This ability to maintain consistent environments is particularly important in scientific computing and bioinformatics, where reproducibility and modularization are critical. Despite these benefits, challenges remain, such as the lack of clear standards for Docker image distribution and potential inconsistencies when different users modify scripts or use different software's versions [165].

There are other containerization tools such as Nextflow [166], Snakemake [167] and Targets [168] that also enable modularization in bioinformatics. Nextflow is a powerful

workflow management system that facilitates the writing and deployment of computational pipelines in a portable and reproducible manner [166]. Built to support the use of Docker containers, Nextflow simplifies the development of complex data analysis workflows by allowing each step to be isolated in a container. This ensures that the software environment remains consistent across runs and platforms. Nextflow's domain-specific language (DSL) is easy to use and flexible, allowing researchers to efficiently manage dynamic workloads and adapt to available computing resources. This makes Nextflow a popular choice in bioinformatics, especially for high-throughput sequencing data analysis, as it improves reproducibility and scalability. Snakemake is another workflow management system designed for reproducible and scalable data analysis [167]. Inspired by the build automation tool Make, Snakemake is specifically tailored to scientific workflows, particularly in bioinformatics. Like Nextflow, Snakemake integrates seamlessly with Docker, allowing each task within a workflow to be executed in a closed environment. This not only ensures reproducibility, but also improves portability across different computing infrastructures. Snakemake automatically manages dependencies and supports parallel execution, making it highly efficient for large-scale computing tasks. Its simplicity and robustness make it a preferred tool for many researchers dealing with complex genomic data analysis. Targets is a workflow framework in R that simplifies the construction and maintenance of data analysis pipelines [168]. This tool allows researchers to define the steps in their analysis pipeline, specifying the relationships between different tasks. Targets uses Docker containers to ensure that each step of the workflow is executed in an isolated and consistent environment. This capability is critical for maintaining reproducibility and ensuring that the data is reproducible.

When choosing a containerization platform for the auto-P2 docking pipeline, Docker stands out for its ease of use and widespread adoption in industry and the scientific community. Docker offers an intuitive and straightforward approach to containerization. Its commands and configuration are relatively easy to learn and use, even for those new to container technologies. Docker Hub, the public repository for Docker images, provides access to a large collection of pre-built images for various applications and tools. This extensive repository allows users to quickly find and use pre-configured images, saving time and effort. While Nextflow, Snakemake and Targets are excellent tools for workflow management, Docker's integration capabilities with these and other workflow systems make it a versatile choice. By choosing Docker for the auto-p2docking pipeline, we are using these advantages to create a flexible and reproducible environment for protein-protein interaction analysis.

Docker images are available for all the tools mentioned above, including FTDOCK, HADDOCK, HDOCK, AF and PDBePisa. Docker images provide a standardized, isolated environment that makes it easy to install and run these complex tools without having to manually configure all the dependencies. Creating a Docker image is a relatively simple process that involves writing a Docker file, specifying the base system and the steps required to install the desired application. This allows Docker images to be easily integrated into automated pipelines, ensuring reproducibility and efficiency in the execution of bioinformatics tasks. The availability of these Docker images for docking, structure prediction and protein interaction analysis tools greatly simplify the implementation of robust and reproducible bioinformatics research and development pipelines such as auto-p2docking.

1.6. Motivation

The determination of PPI sites using traditional experimental methods can be time consuming and resource intensive. *In silico* pipelines of methods, such as docking using structure-based approaches, offer an efficient and less costly method for predicting such sites. This motivated the development of an automated pipeline for the study of PPIs called auto-p2docking.

Auto-p2docking simplifies the traditional analysis of PPIs, which typically involves obtaining 3D structure predictions of the interacting proteins, predicting interacting sites for each protein monomer that can be used in the search for docking orientations of the target protein (receptor) and the ligand. To narrow down the sampling search, predictions of interacting residues for each protein can be used as unbound monomer surfaces. This is done using structure-based methods (which consider solvent accessibility, surface curvature, electrostatic potentials and the secondary structure of the protein) and/or sequence-based methods (which consider evolutionary information, conservation scores, physicochemical properties and sometimes structural features). Then, computational tools for protein-protein docking, where the receptor is placed on a fixed grid and the ligand on a moving grid, such as FTDock [59] and HDOCK [64] (which use FFT approximation for energy evaluation), or a semi-flexible driven approach such as HADDOCK [79] (which implements Ambiguous Interaction Restraints to allow movement of interface residues (active and passive sites) of the two proteins and uses crystallographic and NMR systems for intermolecular energy evaluation). Parameters such as interface area, residue/atom composition and contacts, hydropathy index charge distribution, topological complements as implemented in PISAePDB [85] are then used to identify biologically relevant contacts from the PDB file of the protein complex.

This method has been used to identify PPI regions in proteins such as ATX1 and ATX3, which have many interactors and are involved in neurodegenerative diseases caused by abnormal polyglutamine expansions [169]. Performing such analyses involves submitting data to various online bioinformatics software, waiting for results, and then downloading them from web servers. This manual process is time consuming and error prone. Auto-p2docking is containerized in a Docker image, which has several advantages, including 1) accessibility, promoting open science by making the pipeline available to everyone; 2) simplified setup, so users don't have to install multiple pieces of software, saving time and avoiding compatibility issues; and 3) portability, as Docker images can be easily run on different operating systems and require minimal technical knowledge to install.

Auto-p2docking provides a unified solution that automates PPI interface inference, consolidates the tools required and provides flexibility through a modular architecture that allows a protocol to be customized based on the data available. This makes the methodology less error-prone, more reproducible and accessible to laboratory researchers regardless of their bioinformatics expertise. To address the issue of version control and ensure reproducibility, all versions of the programs and modules used in the analysis performed by the auto-p2docking pipeline are stored in a csv file.

2. Methods

I have developed a modular pipeline called auto-p2docking that provides flexibility for different protein analyses. All modules work individually and can be combined in different ways to customize the protein analysis. The modular design simplifies the maintenance of the pipeline and changes or updates can be made to individual modules without affecting the whole pipeline. The modules are written in Bash, with some additional code written in Python. Typically, the module calls a Docker image to perform the main function of the module. By using Docker images, the modules can ensure consistent and reproducible environments, reducing issues related to software dependencies and configurations. Using Bash and Python with Docker ensures that the modules can run on different systems and environments, improving the interoperability of the pipeline.

Certain modules require specific variables that are specified in a unique configuration file to maintain organization. These variables are prefixed with the module name, such as "haddock_all_files" for the haddock module, so it's easier for the user to keep track of the parameters needed for their analysis. For the modules evoppi_querier and cport_like, variables are stored in individual configuration files to allow greater flexibility. A file called "pipeline" organizes the analysis structure, with each line formatted as "module_name input_folder output_folder". The modules are designed so that the output files from one module are used as the input for the next. However, users can use their own files if they are in the correct format for the module to work.

To prevent users from running the pipeline with errors, a feature has been added to check that the module names in the pipeline file are correct. In addition, a module called pipeline_drawing generates a diagram of the pipeline, allowing users to visually inspect and ensure the correct structure before execution. The pipeline diagram is in SVG format, making it easy to modify and use for visual support in scientific articles.

A feature called "block" has been added to some modules, allowing users to complete the analysis for one ligand at a time, rather than processing each step for all ligands simultaneously. This feature works with a number such as "block 3", which means that the module with the block and the next two will all be run for one ligand at a time, allowing results to be seen more quickly

During pipeline execution, a folder is created to store intermediate files from each analysis step, allowing users to trace how results were generated. Another folder called files_to_keep is created, which contains all the files needed to reproduce the analysis under the same conditions.

Most of the modules use docker images and have been designed to allow the user to choose which version of the docker image to use. This allows for more flexibility and reproducibility in the analysis. Modules will use the latest version when building the module as default.

A .csv file is generated listing all versions of the programs and docker images used in the analysis, along with the relevant references for these programs, making it easier to reproduce the analysis and providing valuable information for use in a scientific article.

I also created an auto-p2docking manual. This feature provides comprehensive descriptions of input/output formats, parameter explanations and usage examples for each module. This documentation ensures that users have immediate access to the information they need, reducing confusion and errors, and promoting efficient use of the pipeline.

Pre-defined protocols have been developed to further enhance the usability and reliability of Auto-p2docking. These protocols provide structured and tested workflows for different types of PPI analyses. Each protocol comes with detailed README files that explain its purpose and provide clear instructions for use, making it easy for researchers to implement them effectively. In addition, the protocols include sample result files that

serve as practical examples of output and help users validate their analyses. By ensuring that all components function correctly and deliver accurate results, these protocols save researchers valuable time and reduce the need for extensive setup and debugging. This structured approach facilitates more efficient and effective scientific research, allowing users to perform sophisticated bioinformatics analyses with confidence.

3. Results

3.1. Building and running the Docker Image

The desire to automate the methodology for inferring PPI interfaces and analyzing their information culminated in the creation of auto-p2docking, a bash script-based pipeline for studying PPIs, which was subsequently packaged in a Docker image for better portability and ease of use. This Docker image runs a multi-software bioinformatics pipeline that allows the identification and characterization of protein-protein interaction interfaces and their properties. It contains 22 modules that have been created with the aim of adding flexibility to the pipeline. I will explain in more detail how to use the pipeline and each module, as well as how to fill the configuration and pipeline files, before explaining the different ways the user can build and run the pipeline, and where the results are located.

The pipeline can be used in any computing environment where Docker is installed, and this Docker image is available on the host machine. At the top of the pegi3 Bioinformatics Docker Images Project website (https://pegi3s.github.io/dockerfiles/) there is a small section explaining how to do this, although instructions are easily found in the Docker guides and all over the internet, as Docker is so widely used.

Before starting the analysis, it is necessary to organize the modules to be used and the input files. As the pipeline is modular, it can be adapted to the user's needs. It can be used with just a geneID of the receptor as input and the rest of the necessary data will be generated by the pipeline, or if the user has input data from other analyses, these can be inserted as input to the modules, but care must be taken to ensure that they are in the correct format and with the correct names. Details of the location of these files and what the correct format is will be explained later. There are two main files required for the pipeline to work: pipeline and configuration files. The pipeline file contains all the modules we want to use, as well as the names of the input and output folders for each module. The configuration file contains the necessary parameters for some modules. Once the pipeline and configuration files and the input folders are in the same directory, we're ready to run the pipeline. To make sure that the pipeline is constructed in the right way, we've created an additional module called pipeline_drawing, which allows you to see a diagram of the pipeline to avoid running it with errors.

After filling the configuration file, building the pipeline and making sure that all input folders are present, the user should run the following command to use the Docker image of the pipeline: "docker run --rm -v $(pwd):/data -v /var/run/docker.sock:/var/run/docker.sock pegi3s/auto-p2docking:1.0.0". Figure 1 shows the pipeline workflow, with all the tasks that can be performed and the modules that make them possible.

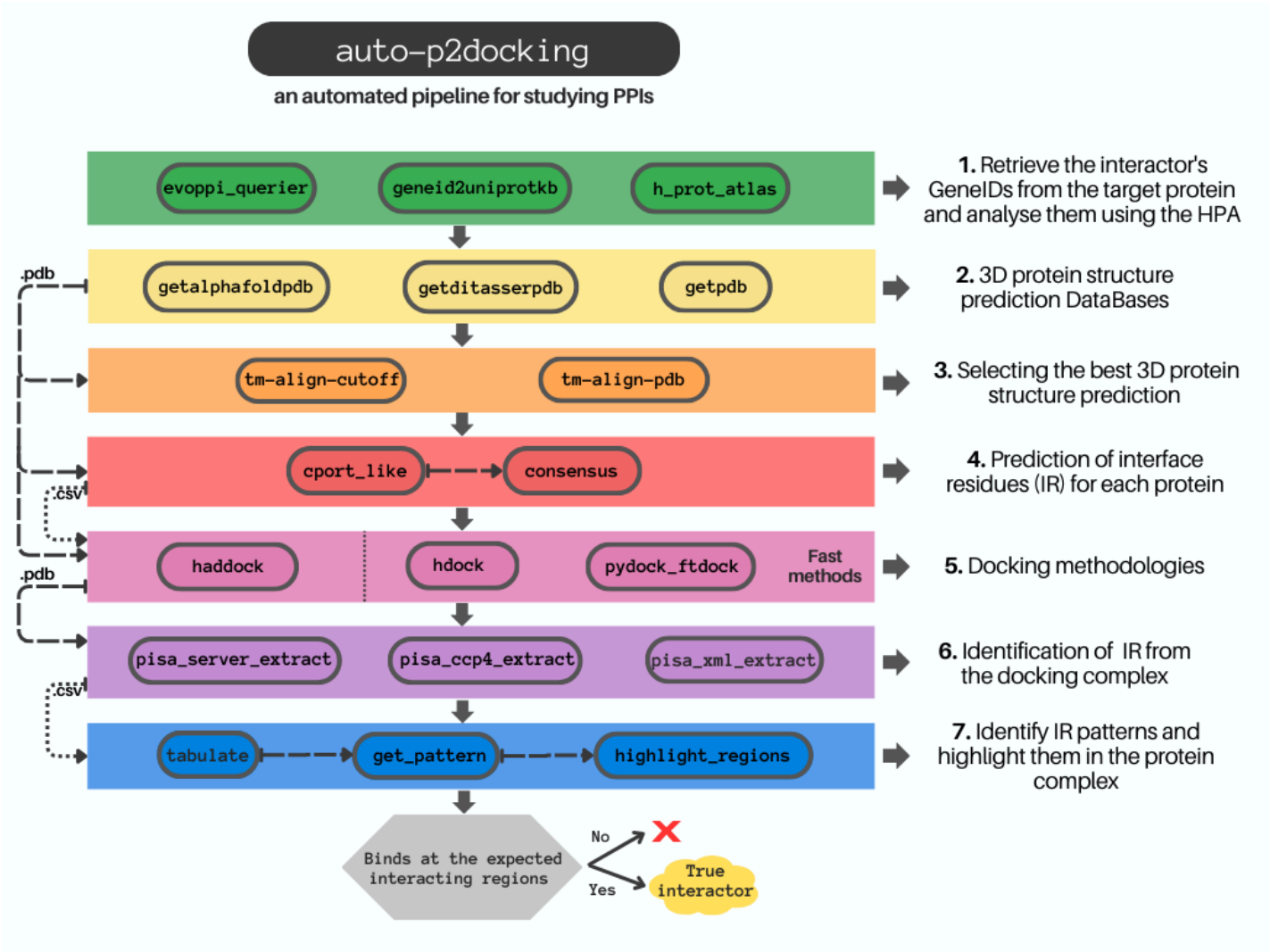

FIGURE 1. Diagram of the pipeline's workflow.

3.2. Modules explanation and parameters

3.2.1. evoppi_querier

This module allows you to retrieve GeneID lists from the EvoPPI databases (see 1.3.1.1. EvoPPi). It takes as input a GeneID and returns from EvoPPI the corresponding list(s) of interactors, as text files with one GeneID per line. The following parameters must be declared, one per line, in a file called config, which should be in the input folder of the module.

Example:

evoppi_q_species="Homo sapiens"
evoppi_q_geneid=411
evoppi_q_level=1
evoppi_q_intdb="BioGRID; HIPPIE; Mentha; Mint; Pina2"
evoppi_q_intmod="Homo sapiens (Modifiers_22)"
evoppi_q_intpolyq="Homo sapiens (PolyQ_22)"
evoppi_q_preddb="Based on Drosophila melanogaster BioGRID (DIOPT); Based on Drosophila melanogaster BioGRID (ENSEMBL)"
evoppi_q_predmod=*
evoppi_q_predpolyq="Homo sapiens Danio rerio (from DIOPT) (PolyQ_models_22)"
evoppi_q_max_int=15

Parameter description:

- evoppi_q_species= indicate the name of the species of interest, that can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_species. In the example provided is: "Homo sapiens"
- evoppi_q_geneid= indicate the GeneID for the gene that encodes the protein to be studied. In the example provided is: 411
- evoppi_q_level= specifying the interaction level (which is the degree of distance (up to a maximum of three) to retrieve transitive interactions). In the example provided is: 1
- evoppi_q_intdb= indicate the name of the interactome databases to be analyzed. The names of the interactomes for the evoppi_q_int.* parameters can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <species> -dt interactome. In the example provided is: "BioGRID; HIPPIE; Mentha; Mint; Pina2"
- evoppi_q_intmod= indicate the name of the Modifiers interactome databases to be analysed. The names of the Modifiers interactomes for the evoppi_q_int.* parameters can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <species> -dt interactome. In the example provided is: "Homo sapiens (Modifiers_22)"
- evoppi_q_intpolyq= indicate the name of the PolyQ interactome databases to be analyzed. The names of the PolyQ interactomes for the evoppi_q_int.* parameters can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <species> -dt interactome. In the example provided is: "Homo sapiens (PolyQ_22)"
- evoppi_q_preddb= indicate the name of the predictomes (predict interactome) databases to be analyzed. The names of the predictomes for the evoppi_q_pred.* parameters can be obtained with the following

command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <Species> -dt predictome. In the example provided is: "Based on Drosophila melanogaster BioGRID (DIOPT); Based on Drosophila melanogaster BioGRID (ENSEMBL)"

- evoppi_q_predmod= indicate the name of the Modifiers predictomes databases to be analyzed. The names of the predictomes for the evoppi_q_pred.* parameters can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <Species> -dt predictome. In the example provided is: "*" (select all available options)
- evoppi_q_predpolyq= indicate the name of the PolyQ predictomes databases to be analyzed. The names of the predictomes for the evoppi_q_pred.* parameters can be obtained with the following command: docker run --rm pegi3s/evoppi-querier list_interactomes --species <Species> -dt predictome. In the example provided is: "Homo sapiens Danio rerio (from DIOPT) (PolyQ_models_22)"
- evoppi_q_max_int= Indicate the maximum number of interactions to be analyzed. In the example provided is: 15

Note 1: all parameters are mandatory, otherwise the program will not run. evoppi_q_int.* and evoppi_q_pred.* can be empty, but at least one interactome/predictome name must be specified. If you want to select all available options, you can use "*" (like evoppi_q_intdb=*).

Note 2: the output folder of EvoPPi accepts up to two output files.

3.2.2. geneid2uniprotkb

This module allows getting the reference UniProtKB number associated with a given GeneID. It accepts a list of Gene IDs, one per line, and returns a list of UniprotKB from NCBI gene (https://www.ncbi.nlm.nih.gov/gene), one per line.

3.2.3. intersect

This module allows you to get the intersection of two or more UniProtKB or GeneID lists. It generates a list of all UniProtKBs or GeneIDs common to all files.

3.2.4. exclude

This module allows you, given two lists of two UniProtKBs/GeneIDs, to get one list with all UniProtKBs/GeneIDs present in the larger list that are not present in the smaller list.

3.2.5. copy

This module copies all files from one folder to another. When copying the .pdb files of the ligands, it's important to check that the output folder name is "PDBs/Ligands". For the receptor .pdb file the output folder should be "PDBs/Receptor".

3.2.6. human_prot_atlas

This module allows retrieving lists of proteins encoded by genes expressed in a given tissue. It accepts as input a list with UniProtKb numbers, one per line, and returns a list of UniProtKb numbers, one per line, of those genes that are expressed in the specified tissue (see 1.3.1.2. The human Protein atlas). In the auto-p2docking configuration file, there are three parameters do be specified.

Example:

h_prot_atlas_inc=*
h_prot_atlas_mode=
h_prot_atlas_exc="Skin; Liver"

Parameter description:

- h_prot_atlas_inc= Specify the tissues to analyse, indicating a list of tissue names separated by ;, or * to analyse all available tissues (that is used by default).
- h_prot_atlas_mode= you can select all proteins by writing union or those that are present in all selected tissues if you write intersection . If you do not provide information in this field, union is used by default. In the example provided is: "", union is used.
- h_prot_atlas_exc= In the case of selecting all tissues (h_prot_atlas_inc= *) you can exclude a specific tissue by indicating it in the name of the tissue(s) to be excluded, separated by ;. In the example provided, Skin and Liver.

The available tissues are: Brain_cerebral_cortex, Brain_hippocampal_formation, Brain_amygdala, Brain_basal_ganglia, Brain_thalamus, Brain_hypothalamus, Brain_midbrain, Brain_cerebellum, Brain_pons, Brain_medulla_oblongata, Brain_spinal_cord, Brain_white_matter, Choroid_plexus, Salivary_gland, Esophagus, Tongue, Stomach, Intestine, Pancreas, Kidney, Urinary_bladder, Breast, Vagina, Cervix, Endometrium, Fallopian_tube, Ovary, Placenta, Skin, Adipose_tissue, Seminal_vesicles, Prostate, Epididymis, Testis, Gallbladder, Liver, Lymphoid_tissue, Bone_marrow, Lung, Pituitary_gland, Thyroid_gland, Parathyroid_gland, Adrenal_gland, Smooth_muscle, Heart, Retina.

3.2.7. getalphafoldpdb

This module allows you to retrieve PDB files from the AF (see 1.3.2. Three-dimensional protein structure prediction) database (https://alphafold.ebi.ac.uk/). It accepts a single text file containing a list of UniprotKB numbers (one per line), located in the input directory, and returns the corresponding files in PDB format from the AF database. The names of the output files are in the following format: UniProtKBnumber_AF.pdb.

3.2.8. getpdb

This module allows retrieving PDB files from the PDB (see 1.3.2. Three-dimensional protein structure prediction) database (https://www.rcsb.org/). It accepts as input a list with UniProtKb numbers, and returns, if available, the PDB files associated with them. The names of the output files are in the following format: UniProtKBnumber_PDBaccession_PDB.pdb.

3.2.9. getditasserpdb

This module allows you to retrieve PDB files from the HPmod (see 1.3.2. Three-dimensional protein structure prediction) database (https://zhanggroup.org/HPmod/). It accepts a single text file containing a list of UniprotKB numbers (one per line), located in the input directory, and returns the corresponding files in PDB format. The names of the output files are in the following format: UniProtKBnumber_ditasser.pdb.

3.2.10. tm-align-pdb

This module allows the automated submission of PDB files obtained from AF, HPmod database, or any other database (file names must not end with the '_PDB' tag) with a PDB file (file names must end with the '_PDB' tag) from the PDB database to the TM-align server (https://zhanggroup.org/TM-align/) (see 1.3.3 Structural Alignment) , in order to select the PDB file with the highest TM score when compared with data from the PDB database. Only PDB files with the same UniProtKB number will be compared. If no data exists in PDB database for the protein being analyzed, no action is taken.

3.2.11. tm-align-cutoff

This module allows automatic submission of pairs of PDB structures (AF, HPmod, or any other database) to the TM align server (https://zhanggroup.org/TM-align/) (see 1.3.3 Structural Alignment), to remove PDB files that have very similar structures (according to the value declared in the variable tm_cutoff_ref). If there is nothing to compare, no action is taken. The file names are not changed. Only file names starting with the same Uniprot number are compared.

Example:

```
tm_cutoff_ref=0.45
tm_cutoff_order="AF-ditasser-itasser"
```

Parameter description:

- tm_cutoff_ref= Specifies the minimum value of the TM score. If the TM-score value is above that declared, then the file that is retained is the one first matching one of the tags declared in the variable tm_cutoff_order=. If the TM-score is below the one specified in the tm_cutoff_ref parameter, both PDB files will be kept. In the example given is 0.45
- tm_cutoff_order= Specifies the order of the files to be kept. In the example provided is "AF-ditasser-itasser" which would select the AF .pdb if the TM-score is above the tm_cutoff_ref parameter.

3.2.12. cport_like

This module uses a similar process to CPORT's algorithm [84] to retrieve active and passive sites for each protein from different servers. The user can choose what servers to use, the available options are SCRIBER [82], SPPIDER [139], PSIVER [83], ISPRED4 [140] and ScanNet [170] (for details see 1.3.4 Prediction of protein Interface Residues).

Example:

```
cport_methods="sppider scannet"
cport_receptor=y
cport_psiverwi= 3600
cport_psivernr=1000
cport_ispred4wi=3600
cport_ispred4nr=1000
cport_scriberwi=60
cport_scribernr=30
cport_spidderwi=60
cport_spiddernr=30
cport_scannetwi=60
cport_scannetnr=30

cport_exclude=receptor
```

Parameter description:

- cport_methods= Specifies the servers to use in cport_like module. In the example given are "sppider scannet"
- cport_receptor=y the methodology is also applied to the receptor protein. If not (n) active and passive sites must be declared.
- cport_psiverwi= wait time (in seconds) between retries to the PSIVER server. In the example given is 3600
- cport_psivernr= number of retries to the PSIVER server. In the example given is 1000
- cport_ispred4wi= wait time (in seconds) between retries to the ISPRED4 server. In the example given is 3600
- cport_ispred4nr= number of retries to the ISPRED4 server. In the example given is 1000
- cport_scriberwi= wait time (in seconds) between retries to the SCRIBER server. In the example given is 60
- cport_scribernr= number of retries to the SCRIBER server. In the example given is 30
- cport_spidderwi= wait time (in seconds) between retries to the SPIDDER server. In the example given is 60
- cport_spiddernr= number of retries to the SPIDDER server. In the example given is 30
- cport_scannetwi= wait time (in seconds) between retries to the ScanNet server. In the example given is 60
- cport_scannetnr= number of retries to the ScanNet server. In the example given is 30
- cport_exclude= Specifies what to exclude from the analysis. If it's equal to "ligands", the active and passive sites will be calculated only for the receptor, and vice versa. If it's empty, both ligands and receptor will be calculated. In the example given is "receptor"

Note 1: all these parameters must be in an independent config file inside the cport_like input folder. The only parameter that needs to be placed in the global config file it's "cport_exclude=".

Note 2: To create the 'cat' (categories) files the following rules were used (0,1 and 2 means undetermined, active and passive sites, respectively) :

- ISPRED4: values equal or above '0.5' are replaced by '1', less than '0.5' are replaced by '2', and '-' are replaced by '0'.
- SCRIBER: values equal or above '0.5' are replaced by '1', below '0.27' are replaced by '2', and in between '0.27' (including) and '0.5' by '0'. Missing positions are replaced by '0'.
- SPPIDER: 'A' has been replaced by '1' and '-' by '0'.
- PSIVER: values equal or above '0.5' are replaced by '1', equal or less than '0.37' are replaced by '2', and in between '0.37' and '0.5' by '0'. Missing positions are replaced by '0'.
- ScanNet: values equal or above '0.5' are replaced by '1', equal or less than '0.35' are replaced by '2', and in between '0,35' and '0.5' by '0'. Missing positions are replaced by '0'.

Note 3: If this module is used alone in a single module pipeline, it's necessary to use the copy module to copy all the necessary PDBs. For example, if the user wishes to calculate the active and passive sites for a ligand and a receptor, the respective PDBs must be in different input folders in the working_dir. The copy module should then be used to copy the files to PDBs/Ligands and PDBs/Receptor respectively. Like the following example, where the pdb file for the ligand it's in input1, the pdb for the receptor it's in input2 and the config file it's in input3.

Example

```
copy input1 PDBs/Ligands
copy input2  PDBs/Receptor
cport_like input3 output1
```

3.2.13. consensus

This module calculates the consensus of the methods selected in cport_like. The active sites are marked as 1, the passive sites as 2 and the undetermined sites as 0. The choice of active and passive sites was based on the following rules when counting all options for a site in the files of the different methods analyzed:

- If the number of 0 was greater than the number of 1 and the number of 0 was greater than the number of 2, the site was considered as 0 (undetermined).
- If the number of 1 was greater than or equal to the number of 0 and the number of 1 was greater than the number of 2, the site was considered to be 1 (active).
- If the number of 2 was greater than or equal to the number of 0 and the number of 2 was greater than the number of 1, the site was considered to be 2 (passive).

Then, if the number of active sites is lower than that specified in consensus_active_min_number then a single method is chosen to create the list of active and passive sites. The chosen method is the one showing the highest number of active sites. If, as the result of this operation, there are more active sites than the number specified in consensus_active_max_number, the below conditions are applied as well.

If in the consensus there are more active sites than the ones specified in consensus_active_max_number, then a new consensus is calculated based on two methods only (those specified in consensus_preferred_1 and consensus_preferred_2). If this option still produces more active sites than the number specified in consensus_active_max_number, then, if there are two consecutive active sites, the

second one is not considered. If this procedure still produces more active sites than the ones specified in consensus_active_max_number, then a random set of active sites is chosen from the consensus produced in the previous step, to fulfil the condition imposed by consensus_active_max_number.

It should be noted that if the conditions imposed by consensus_active_min_number and consensus_active_max_number is not met based on the original consensus procedure, and if the cport_like and consensus modules are invocated again in the pipeline, the whole procedure will be repeated based on the previous methods plus the new ones (please note that the results of the previous cport-like invocation are used). If the condition is, however, met, no action is taken when invoking again the cport_like and consensus modules.

If there are fewer active sites than the ones specified in consensus_critical_active_number, then, the ligand being processed is no longer considered when running the remaining modules of the pipeline.

Example:

```
consensus_preferred_1=psiver.csv
consensus_preferred_2=scriber.csv
consensus_active_max_number=150
consensus_active_min_number= 10
consensus_critical_active_number=0
```

Parameter description:

- consensus_preferred_1= specify the first preferred method to use when the number of active sites is above the maximum number. In the example given is psiver.csv
- consensus_preferred_2= specify the second preferred method to use when the number of active sites is above the maximum number. In the example given is scriber.csv
- consensus_active_max_number= indicate the maximum number of active sites. In the example given is 150
- consensus_active_min_number= indicate the minimal number of active sites. In the example given is 10
- consensus_critical_active_number= indicate the critical number of active sites. If the number of active sites is below or equal this number, the pipeline will not run for the current ligand. In the example given is 0

3.2.14. hdock

This module allows the use of HDOCK (see 1.3.5.1 hdock), a hybrid docking protocol that combines both template-based and template-free approaches. It takes two .pdb files as input and returns the .pdb of the 100 highest scoring docking protein complexes generated by HDOCK. The input .pdb files must be placed in the input folder of the module.

3.2.15. haddock

This module allows the use of HADDOCK, an information-driven flexible docking approach that accurately models the 3D structure of macromolecular complexes (see 1.3.5.3 HADDOCK). This module takes a total of 6 files as input: two protein .pdb files, two active site files (one for each protein) and two passive site files (one for each protein). All 6 files must be placed in the input folder of the module. It returns the .pdb of the de-docking protein complexes with a z-score less than or equal to 0. The complex clusters resulting from all possible predicted dockings are scored based on a statistic called the z-score. The z-score represents how many standard deviations the HADDOCK score of a given cluster is from the mean of all clusters obtained, the lower the z-score, the better the predictions.

Example:

haddock_all_files=n

Parameter description:

- haddock_all_files= If the user wants to see all the files created during the docking process set this parameter to y (yes), if the user only wants to see the most important files, set this parameter to n (no). In the example given is n

3.2.16. pydock_ftdock

This module takes a total of 6 files as input: two protein .pdb files, two files with the active sites (one for each protein) and two files with the passive sites (one for each protein). All 6 files must be placed in the input folder of the module. It returns the .pdb of the docking protein complexes with the highest score generated by pydock (see 1.3.5.2 pydock_ftdock).

Example:

pydock_ftdock_number_of_solutions=100
pydock_ftdock_best_pdb=5

Parameter description:

- pydock_ftdock_number_of_solutions= specify the number of solutions to analyze, this number must be less than or equal to 10000 (recommended: 100). In the example given is 100
- pydock_ftdock_best_pdb= specify the number of best solutions (recommended: 5). In the example given is 5

3.2.17. pisa_ccp4_extract

This module takes as input the .pdb files of protein complexes obtained with the selected docking methodology and returns a .tsv file of the best protein complex (highest number of interacting residues) and the corresponding .pdb file of the best complex, using CCP4 software (1.3.6. Protein Structure analysis).

3.2.18. pisa_server_extract

This module takes as input the .pdb files of protein complexes obtained with the selected docking methodology and returns a summary of the best protein complex (highest number of interacting residues) and the corresponding .pdb file of the best complex, using on-line PDBePISA software (1.3.6. Protein Structure analysis).

3.2.19. pisa_xml_extract

This module takes as input one or more xml files of protein complexes generated by the PDBePISA server and returns a summary (.tsv file) of the best protein complex (highest number of interacting residues). To use this utility with more than one protein pair, please create a main input folder and create as many folders inside it as there are protein pairs to be analyzed. Each folder should contain one or more PISAePDB xml files. This should be used, if you have XML files from the PDBePISA server.

3.2.20. tabulate

This module takes as input .tsv files (generated by the pisa_server_extract, pisa_ccp4_extract, pisa_xml_extract modules) and creates a table with all the ligand data present in the input folder.

3.2.21. get_pattern

This module takes as input the table created in the tabulate module and returns a file named pattern_regions.txt with the pattern positions for structure 1 according to the rule and the cutoff value specified in the auto-p2docking configuration file.

Example:

pattern_cutoff=50
pattern=“111[01]1”

Parameter description:

- pattern_cutoff= specify the percentage above which the site is assigned as "1" in the rule pattern. In the example given is 50
- pattern= specify the pattern rule. In the example given is “111[01]1”, which means that a region is defined as a sequence of at least three sites with more than 50% frequency, which can be separated by a site with less than 50% frequency if the next site is also present in more than 50% of the cases.

3.2.22. highlight_regions

This module applies colors to the PDB structure to highlight regions of interest, based on the pattern_regions.txt file created by the get_pattern module. Chain A (receptor) and B (ligand) are colored green and blue respectively, and amino acid sites of the pattern are colored red. This module also uses pymol (https://pymol.org/) to save a png file of the colored structure. The output directory will contain the colored .pdb files and the corresponding .png files.

3.3. The configuration and pipeline files

The configuration file contains the parameters of the modules (explained in section 3.2), except for evoppi_querier and cport_like. This file must be placed in the folder where the user wishes to run the pipeline, as well as the pipeline file and any input folders. Figure 2 shows an example of a configuration file. The commented lines are the names of the modules and below them are the parameters needed to run the module. In this file it's also set the directory where the pipeline will run, and the name of the project chosen by the user.

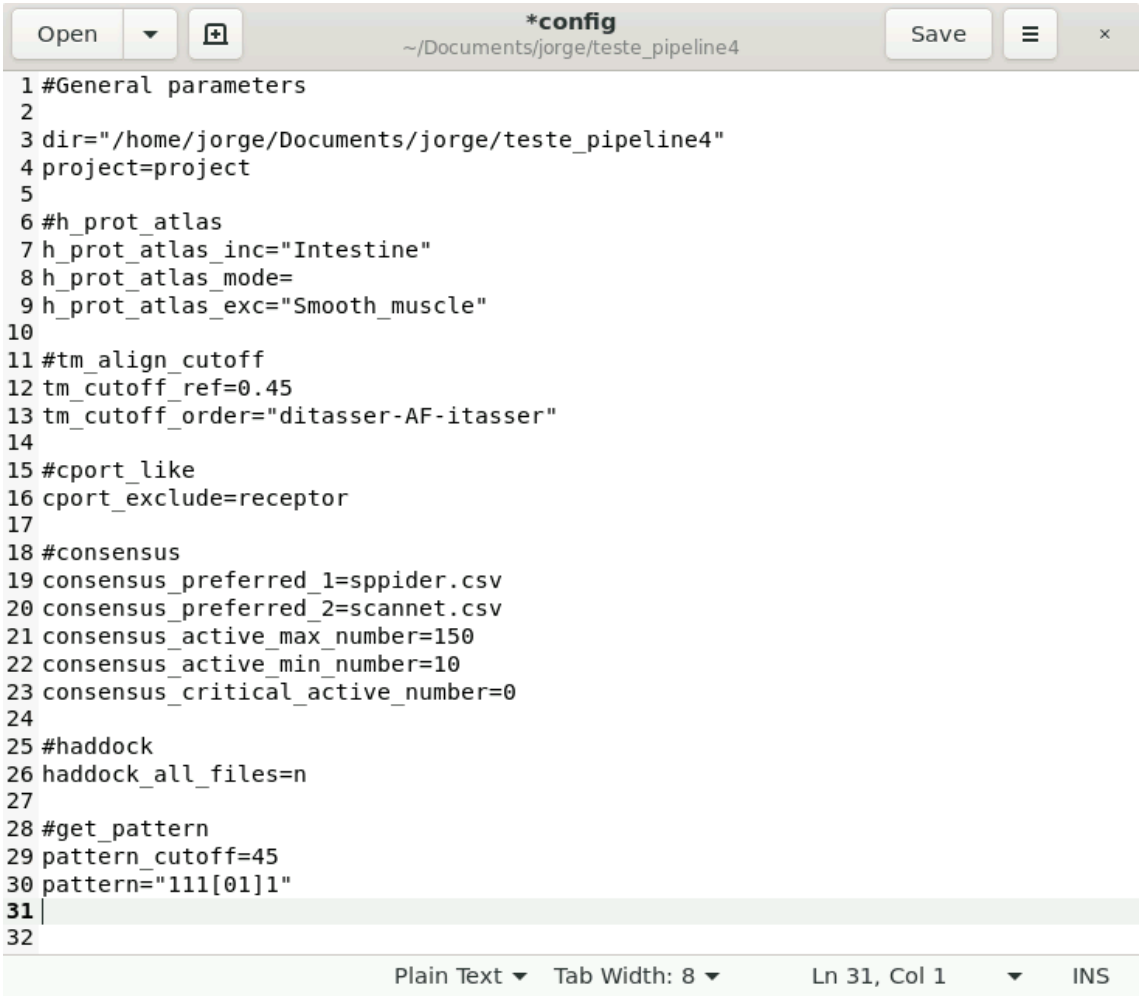

FIGURE 2. Example of a configuration file. The lines starting with "#" are comments. Line 3 is the directory where the pipeline will run and line 4 is the name of the project, the user can choose any name. After that, the comment lines are the names of the modules and below are the necessary parameters. All variables must have the correct name.

Figure 3 shows an example of a pipeline file. Each line can be split into up to five fields. The first is the name of the module, the second is the name of the module's input folder, and the third is the name of the module's output folder (where the results will appear). The first three are present in all modules. The fourth (block) and fifth (number) fields can only be used in the following modules: cport_like, consensus, haddock, pydock_ftdock, pisa_server_extract and ccp4_server_extract. The pipeline was made with the principle that the output of one module would serve as the input for the next module. The process starts with retrieving the interactor's geneIDs of the target protein using evoppi_querier. The selected geneIDs are converted to UniProtKB numbers, and

then the interactors are parsed according to the tissues in which they exist. For the proteins selected, the 3D protein structure prediction of the interactors and the target protein are downloaded, to select the best 3D protein structure prediction. In the example presented in Figure 3, I download interactors 3D structures from Hmod and AF databases, using the getditasserpdb and getalphafoldpdb modules. To select the proteins structures to be used, I used tm-align-cutoff to compare both structures for the same ligand. If the TM score is above the value specified by the user in the tm_align_cutoff parameter, only one file is kept based on the user preference specified in the order parameter. Then the ligand list it's used to download the corresponding data from the PDB. The module tm-align-pdb is used to keep the pdb with the highest TM score compared to the data from the PDB database. The final ligand pdb files are copied into the PDBs/Ligands folder (this is required for the pipeline to work). The same modules are performed for the receptor (download the 3D structures using the getditasserpdb and getalphafoldpdb modules, compare both structures using tm-align-cutoff, download the corresponding data from the PDB, and use tm-align-pdb to keep the pdb with the highest TM score when compared to data from the PDB database) . The final receptor pdb file is copied to the PDBs/Receptor folder. This folder must exist for the pipeline to work.

According to the docking methodology used, it may be necessary to predict interface residues for each protein. For that I used cport-like (Figure 3) to calculate active and passive sites. For the receptor, the same approach can also be performed, by indicating in the cport_receptor parameter. This module retrieves the active and passive sites using the algorithms selected in the cport_methods parameter. The consensus module then makes a consensus of the sites predicted by the selected methods. If a consensus can't be reached, once the cport_like module is called again with the method selected in the new config file present in the cport_like second input folder. Then the consensus module is used again with all the predicted sites to reach a new consensus. The consensus module makes a consensus of the sites predicted by the selected methods. If a consensus can't be reached, the cport_like module is called again with the method selected in cport_additional_methods, then the consensus module is used again with all the predicted sites to reach a new consensus.

From these analyses I get: the receptor and ligand PDBs and the active and passive sites for each protein. Docking of each interactor with the receptor will be performed using the selected methodology. As shown in Figure 3, the haddock module is used. If the user wants to see all the files produced, they must set the haddock_all_files to "y" (the files will appear in the intermediate_files folder). The different solutions of docking, will be used to select the best protein complex structure, using the pisa_server_extract module, and from that complex, the interacting regions of the two proteins are scored.

If the user wants to use a list of active and passive sites for the receptor (Figure 3), the user can select in cport_like in command cport_exclude=receptor. In this case a list of active and passive sites must be placed in the docking method input folder. In the case the user wants to run a set of modules as if it were a single module it can use the special block command. This command can be invoked after the specification of the name of the output folder at the following commands: cport-like, consensus, and docking modules. The number after the word “block” identifies the number of commands that will behave as a block, including the line containing the block command. In Figure 3, “block 6” indicates that cport_like, consensus, cport_like, consensus, haddock and pisa_server_extract, will be run for one ligand at a time. In this way the user can see the docking results as they are obtained for each ligand.

After all the ligands have gone through the block, the tabulate module will organize all the results that came from the pisa_server_extract in a table that will be used by the get_pattern. The get_pattern module will use all the results to calculate the interface residues based on the percentage (pattern_cutoff) of times they appear in all the results, and then see if the interface residues make up the chosen pattern (pattern)

and at what positions. Finally, the highlight_regions module will color the protein complex PDB, receptor and ligand in green and blue respectively, and pattern sites will be colored red based on the patterns found in the get_pattern module. Pymol is used to save a .png of the colored structure.

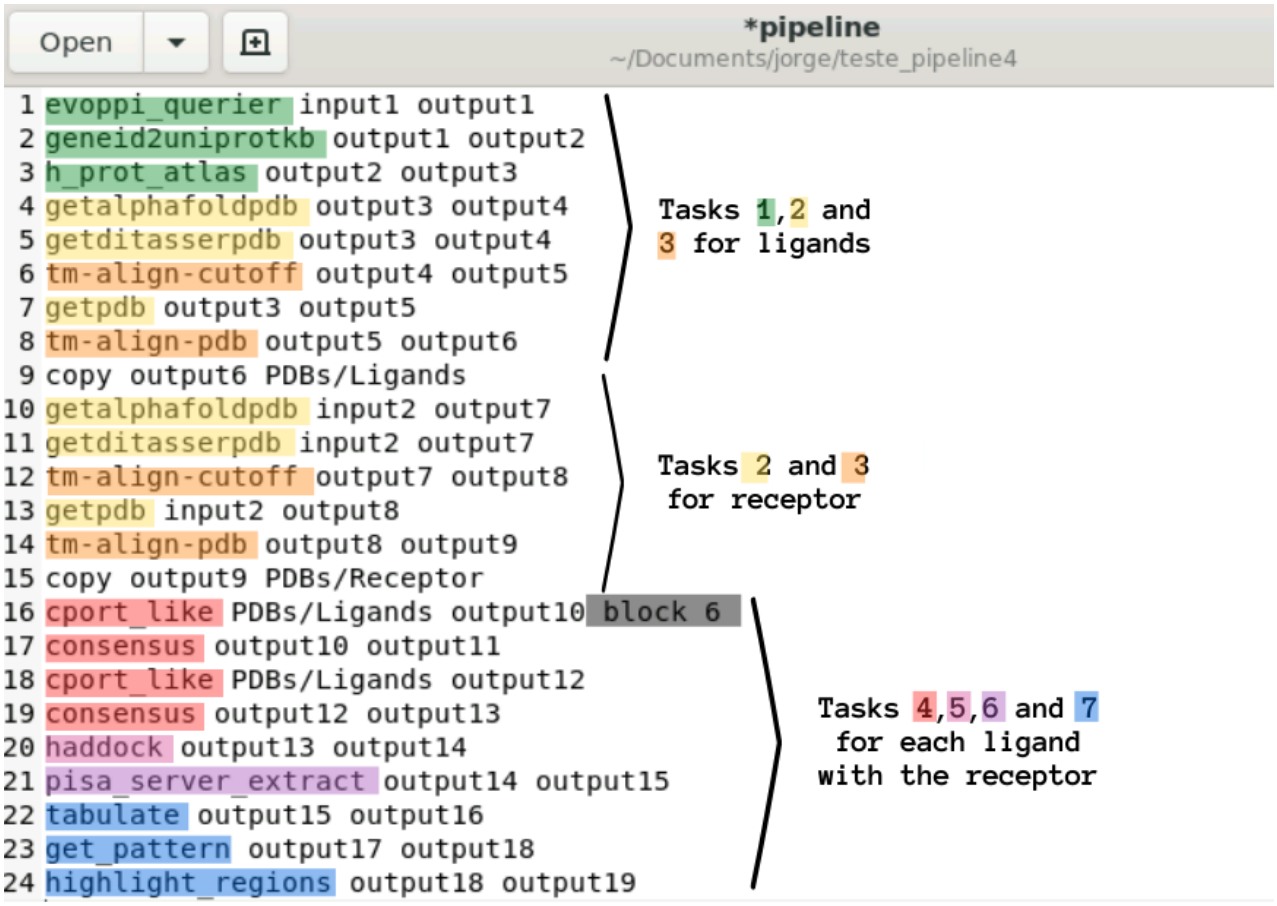

FIGURE 3- Detailed annotation of a pipeline file example, according to the Tasks (1 to 7 marked in green, yellow, orange, red, pink, purple and blue respectively) performed to identify the interacting regions in PPI. In the pipeline file, each line can be divided into up to five fields. The first is the name of the module (that is colored according to the Task it belongs to), the second is the name of the module's input folder, and the third is the name of the module's output folder. These are present in all modules of the pipeline. The fourth (block) and fifth (number) fields can only be used in with following modules: cport_like, consensus, haddock, pydock_ftdock, pisa_server_extract and ccp4_server_extract, allowing these modules to be run for one ligand at a time, and thus allowing the user to obtain the docking results as soon as they are obtained for each ligand.

3.4. Customization of the analysis

The user may have experimental data that justifies skipping some steps in the pipeline and using one type of file instead of another. In auto-p2docking, the user can provide their own files as input for some modules, if they are in the correct format and in the folder selected as input for the module. In this section I will explain the correct format of input files for each module and where to place them.

3.4.1.geneid2uniprotkb

This module takes as input a file with a .gi extension and containing one gene id per line.

3.4.2.human_prot_atlas

This module takes as input a file with a UniProtKB number declared on each line.

3.4.3. intersect

This module takes two UniProtKB or GeneID lists.

3.4.4. exclude

This module takes two UniProtKB or GeneID lists, where one is smaller than the other.

3.4.5.getalphafoldpdb

Takes as input a file where each UniProtKB number is declared on a different line.

3.4.6.getpdb

This module takes as input a file where each UniProtKB number is declared on a different line.

3.4.7.getditasserpdb

This module takes as input a file where each UniProtKB number is declared on a different line.

3.4.8.tm-align-pdb

This module takes as input pairs of .pdb files (which must have the same UniProtKB number). The files must have the following name structure UniProtKB_database (e.g.: P15848_AF.pdb, P15848_ditasser.pdb).

3.4.9.tm-align-cutoff

This module takes as input pairs of .pdb files (which must have the same UniProtKB number). The files must have the following name structure UniProtKB_database (e.g.: P15848_AF.pdb, P15848_ditasser.pdb).

3.4.10. copy

This module copies all files from one folder to another. For the pipeline to work, all ligands and receptors must be in a folder called PDBs, under the project folder (the variable project it is assigned in the config file). When copying the ligands .pdb files it's important to make sure that the output folder is PDBs/Ligands and for the receptor it's PDBs/Receptor.

3.4.11. cport_like

This module accepts .pdb files as input. The files must have the following UniProtKB_database name structure (e.g.: P59665_AF.pdb, P15848_ditasser.pdb).

3.4.12. consensus

This module takes as input the .csv files, which should have the following naming structure UniProtKB_database_server (e.g.: P15848_AF_scannet.csv, P15848_AF_psiver.csv). All .csv files corresponding to proteins should be placed in a folder named UniProtKB_database (e.g.: P15848_AF) inside the module's input folder.

3.4.13. haddock

This module takes as input the .pdb files of the ligand and receptor, which should have the following naming structure UniProtKB_database (e.g. P15848_AF.pdb). It also uses the active and passive sites of each protein, which must have the following name structure: UniProtKB_database_active.sites and UniProtKB_database_passive.sites (e.g.: P15848_AF_active.sites.csv, P15848_AF_passive.sites.csv). The active and passive sites must be placed in a folder named UniProtKB_database (e.g.: P15848_AF) within the module's input folder. The module expects the .pdb files of the corresponding protein to be placed in the PDBs/Ligands and PDBs/Receptor folders.

3.4.14. hdock

This module takes as input the .pdb files of the ligand and receptor, which should have the following name structure UniProtKB_database (e.g. P59665_AF.pdb, P15848_ditasser.pdb). The module expects the .pdb files of the corresponding protein to be placed in the PDBs/Ligands and PDBs/Receptor folders.

3.4.15. pydock_ftdock

This module takes as input the .pdb files of the ligand and receptor, which should have the following naming structure UniProtKB_database (e.g. P15848_AF.pdb). It also uses the active and passive sites of each protein, which must have the following name structure: UniProtKB_database_active.sites and UniProtKB_database_passive.sites (e.g.: P15848_AF_active.sites.csv, P15848_AF_passive.sites.csv). The active and passive sites must be placed in a folder named UniProtKB_database (e.g.: P15848_AF) within the module's input folder. The module expects the .pdb files of the corresponding protein to be placed in the PDBs/Ligands and PDBs/Receptor folders.

3.4.16. pisa_ccp4_extract

This module takes as input the protein complex .pdb files produced by the docking programs. The file names should be in the format ReceptorUniProtKB_database_LigandUniProtKB_database_DockingMethod_name_ ComplexName (e.g.: P15848_ AF_P59665_ditasser_FTDOCK_56.pdb). The files must be placed in a folder named LigandUniProtKB_database (e.g.: P59665_ditasser) within the module input folder.

3.4.17. pisa_server_extract

This module takes as input the protein complex .pdb files returned by the docking programs. The file names should have the following format: ReceptorUniProtKB_database_LigandUniProtKB_database_DockingMethod_name_ ComplexName (e.g.: P15848_ AF_P59665_ditasser_FTDOCK_56.pdb). The files must be placed in a folder named LigandUniProtKB_database (e.g.: P59665_ditasser) within the input folder of the module.

3.4.18. pisa_xml_extract

This module takes as input one or more xml files of protein complexes resulting from PDBePisa. The files must be placed in a folder named LigandUniProtKB_database (e.g.: P59665_ditasser) inside the module's input folder.

3.4.19. tabulate

This module takes as input .tsv files (generated by the pisa_server_extract, pisa_ccp4_extract, pisa_xml_extract modules), these files must be organized in subfolders named after the respective ligand (e.g.: P59665_ditasser) within the input folder of the module.

3.4.20. get_pattern

This module takes as input the table created in the tabulate module. The file must be placed in the input folder of the module.

3.4.21. highlight_regions

This module takes as input the pattern_regions.txt file created in the get_pattern module and the .pdb files to be colored. The pattern_regions.txt file must be placed in the input folder of the module. The .pdb files to be colored must be placed in a folder called files_to_keep, inside a folder called pisa_extract and then organized in subfolders named after the respective ligand (e.g.: P59665_ditasser) (if this module is used in a pipeline that has a docking method and pisa_server_extract or pisa_ccp4_extract, the .pdb files will automatically be placed in the files_to_keep folder).
This module works for any type of pdb, if the structure to be colored is chain A.

3.4.22. Version customization

It is possible to change the version of the docker image used in each module. To do this, simply add the appropriate parameter from the list below to the config file with the desired version. The values assigned to each of the following parameters are the current default versions.

Parameters:

- cport_like_docker_c_version=2024.05
- evoppi_docker_ c_version=1.0.1
- geneid2uniprotkb_docker_ c_version=1.1.0
- getalphafoldpdb_docker_ c_version=1.0.0
- getditasserpdb_docker_ c_version=1.0.1
- getpdb_docker_ c_version=1.0.0
- haddock_docker_ c_version=2.4
- hdock_docker_ c_version=1.1
- highlightregions_docker_ c_version=1.0.0
- hprotatlas_docker_ c_version=1.0.0
- pisaccp4_docker_ c_version=7.0.1
- pisaserver_docker_ c_version=1.0.0
- pisaxmlextract_docker_ c_version=0.22.2
- pydock_program_ c_version=3.2.3
- tmaligncutoff_docker_ c_version=1.0.0
- tmalignpdb_docker_ c_version=1.0.0

3.5. Results and their location

Regardless of the pipeline built and the user configurations, the final results are always saved in a folder with the name assigned by the user to the "project" variable in the configuration file. The main folder (Figure 4) is in the directory selected in the "dir" variable in the configuration file. This folder contains the configuration file, the pipeline file, the necessary input folders, the pipeline.gv.svg, (that is created by the pipeline_drawing module), the "project" folder (Figure 5) that contains all the results, that are created during the analysis according to the name of the "project=" variable in the configuration file (in this case it's also project), and the files_to_keep folder (Figure 8) where the relevant data is kept, as shown in Figure 4. The "project" folder contains all the output folders (where the results of the respective module are stored), the intermediate_files folder folder (where the intermediate files of the analysis are stored), and the PDBs folder (with all .pdb files), as shown in Figure 5. In the "intermediate_files" folder (Figure 6) the blocks_and_commands file (Figure 7) that indicate the data that is included in each of the subdirectories assigned as BxCx, that contains all the intermediate files of each module or block of modules can also be found.

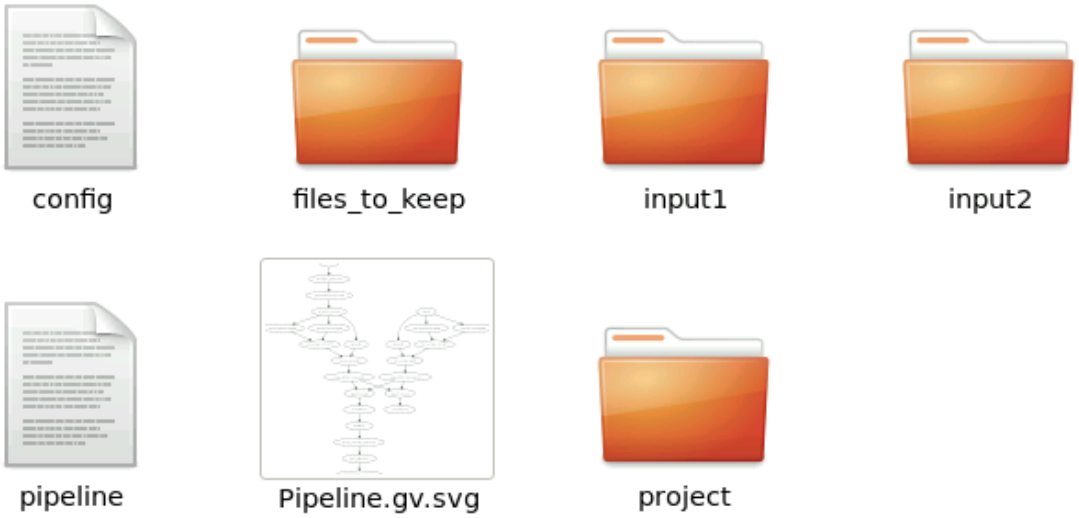

FIGURE 4. Main folder organization: the configuration file, the pipeline file, the necessary input folders, the pipeline.gv.svg, (that is created by the pipeline_drawing module), the "project" folder that is created during the analysis according to the name of the "project=" variable in the configuration file (in this case it's also project) that contains all the output folders, the intermediate_files folder and the PDBs folder, and the files_to_keep folder.

FIGURE 5. "Project" folder organization. It contains all output folders (where the results of the respective module are stored), the intermediate_files folder (where the intermediate files of the analysis are stored) and the PDBs folder (where the ligand and receptor folders with the respective .pdb files are stored).

FIGURE 6. Intermediate_files folder organization, where the intermediate files of the analysis are stored. The “blocks_and_commands” file indicate which subfolder contains the files of each module. The subfolders are organized by steps of the analysis and within them are the intermediate files of each step.

Open | blocks_and_commands | ~/Documents/jorge/teste_pipeline4/project/intermediat... | Save

```
 1 evoppi_querier input1 output1 B1C1
 2 geneid2uniprotkb output1 output2 B1C2
 3 h_prot_atlas output2 output3 B1C3
 4 getalphafoldpdb output3 output4 B1C4
 5 getditasserpdb output3 output4 B2C1
 6 tm-align-cutoff output4 output5 B2C2
 7 getpdb output3 output5 B3C1
 8 tm-align-pdb output5 output6 B3C2
 9 copy output6 PDBs/Ligands B3C3
10 getalphafoldpdb input2 output7 B4C1
11 getditasserpdb input2 output7 B5C1
12 tm-align-cutoff output7 output8 B5C2
13 getpdb input2 output8 B6C1
14 tm-align-pdb output8 output9 B6C2
15 copy output9 PDBs/Receptor B6C3
16 cport_like project/PDBs/Ligands project/output10 && consensus project/-
   output10 project/output11 && cport_like project/PDBs/Ligands project/-
   output12 && consensus project/output12 project/output13 && haddock project/-
   output13 project/output14 && pisa_server_extract project/output14 project/-
   output15 B7C1
17 tabulate output15 output16 B8C1
18 get_pattern output16 output17 B8C2
19 highlight_regions output17 output18 B8C3
20
```

FIGURE 7. Blocks_and_commands file. In this file we can see in which subfolder the intermediate files of each step of the analysis have been placed.

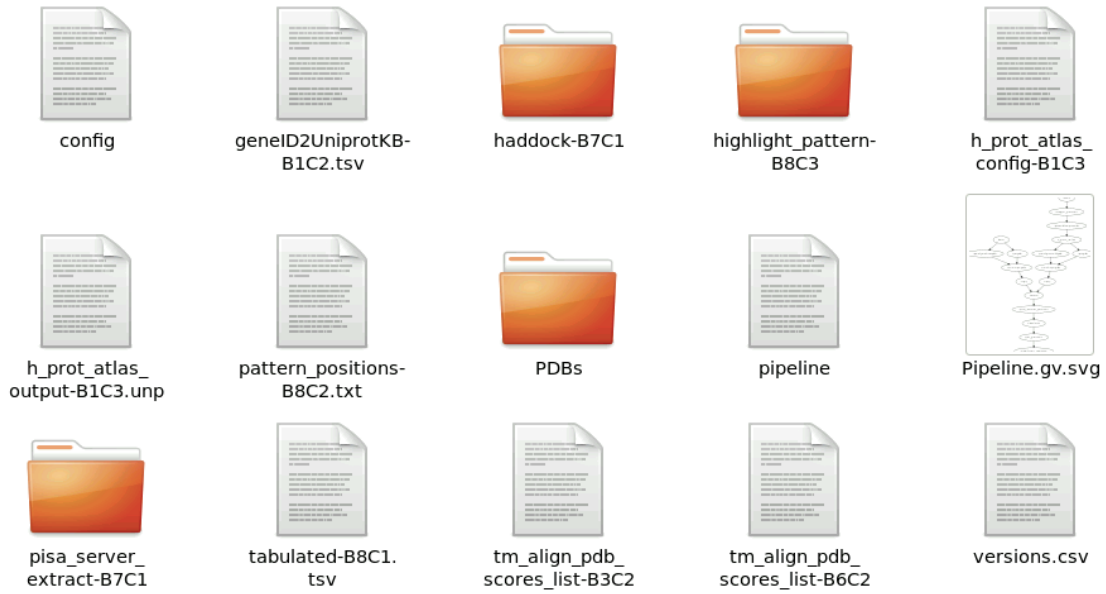

FIGURE 8. Files_to_keep folder. This folder contains all the files necessary for the user to repeat the analysis under the same conditions. It stores the output of the geneid2uniprotkb module with the UniprotKB selected to start the analysis, the h_prot_atlas configuration file and the output file, then the lists of TM score comparisons made by the tm_align_pdb module for the ligand(s) and for the receptor. It saves all the protein complex pbd files generated by HADDOCK (in this case) and the output of pisa_server_extract. From the tabulate module the output table is saved, for get_pattern the pattern file and for highlight_regions the png saved from the pymol and the pdb of the protein complex.

3.6. Protocols

The pre-defined protocols in auto-p2docking provide a step-by-step guide; these protocols ensure that complex workflows are carried out in a consistent manner. Each protocol is accompanied by a detailed README file that explains its purpose and provides clear instructions on how to use it. This documentation ensures that users can easily understand and implement the protocols regardless of their level of expertise. In addition, each protocol includes sample result files that serve as practical examples of output and help users validate their analyses. The protocols use thoroughly tested and validated modules, ensuring that all components function correctly. This structured approach not only saves researchers valuable time, but also allows them to perform bioinformatics analyses with confidence. By reducing the need for extensive setup and debugging, the predefined protocols in auto-p2docking enable more efficient and effective scientific research.

Inside each protocol folder are all the files and folders necessary to perform the analysis. For example, in Figure 9, the folder for protocol1 contains the pipeline file, the config file, the necessary input folders, the pipeline representation in SVG format, the README file with the protocol explanation and usage instructions, and a zip folder called 'execution'. This zip folder contains the project and files_to_keep folders of the protocol, as well as the log file and execution time (in a high-performance computer with 256 GB of RAM with 72 cores, Intel Xeon(R) CPU ES-2695 v4 Core Processor). This comprehensive set of files ensures that users have all the resources they need to run the protocol without a problem.

Note: The execution time will depend on the computer used, and the results of the protocols may not be exactly the same since the databases are continuously updated and modules like EvoPPi use random functions when necessary.

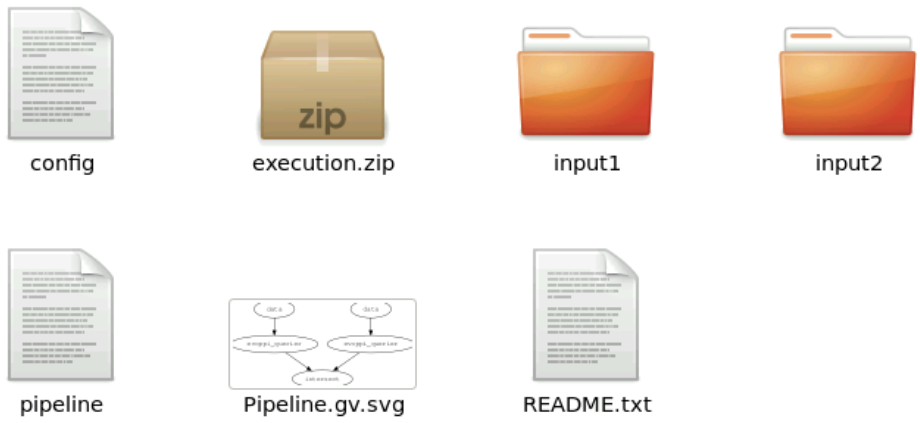

FIGURE 9. Protocol_1 folder. It contains all the files and folders needed to run the protocol, plus a zip folder containing the results and execution time.

3.6.1. Protocol 1

Is suitable for comparing data from different EvoPPI databases. In this protocol, the intersection of the EvoPPI databases (main databases) and the Predictome databases produces a list of GeneID numbers for genes encoding the interactors of the protein encoded by the specified gene, reported in proteomic analyses from several species (according to the selected Predictome databases). Figure 11 shows a graphical representation of the analysis.

Modules used: evoppi_querier and intersect (Figure 10)
Execution time: 4 minutes and 5 seconds

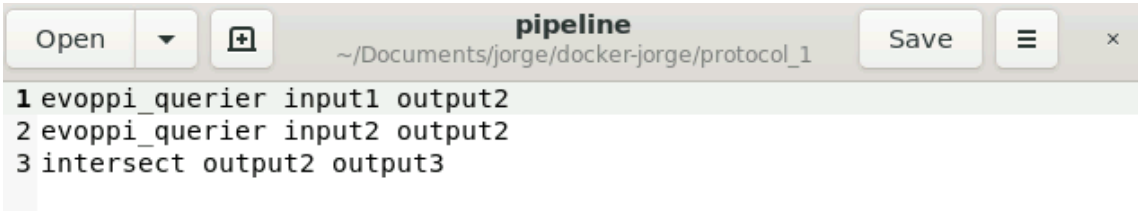

FIGURE 10. Pipeline file of protocol_1.

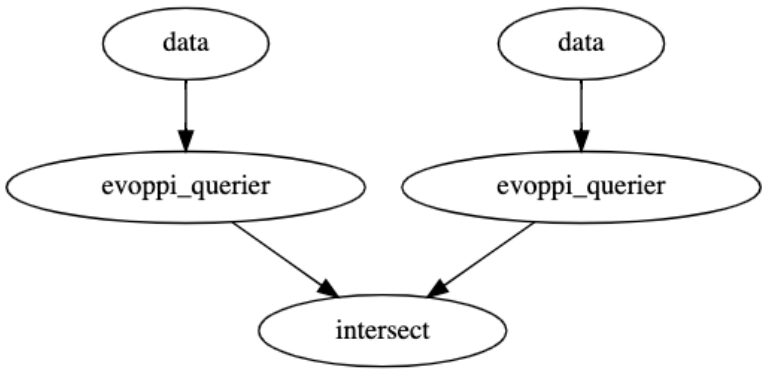

FIGURE 11. Graphical representation of the analysis performed on protocol_1.

3.6.2. Protocol 2

Is suitable for avoiding duplication of analyses. In this protocol, the user provides a list of GeneID or UniProtKb numbers to be ignored when generating a new list of GeneID or UniProtKb numbers to be used in further analyses. The file provided by the user is copied to a folder containing a file with the same type of data, and the accession numbers present in the smaller file (which should be the one provided by the user) are deleted from the larger file (which should be the complete list of accession numbers to be considered). Figure 13 shows a graphical representation of the analysis.

Modules used: evoppi_querier, copy and exclude (Figure 12)
Execution time: 1 minute and 21 seconds

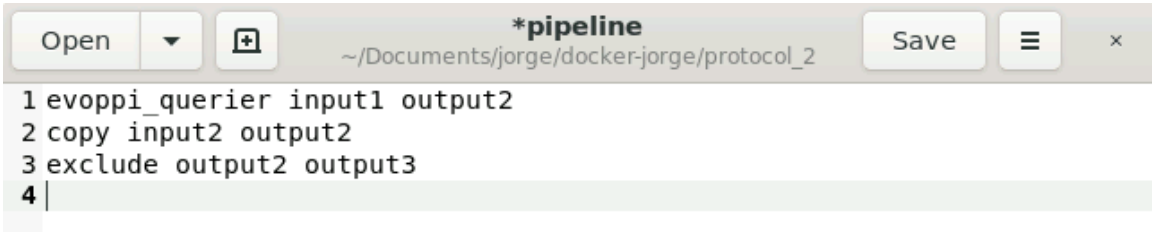

FIGURE 12. Pipeline file of protocol_2.

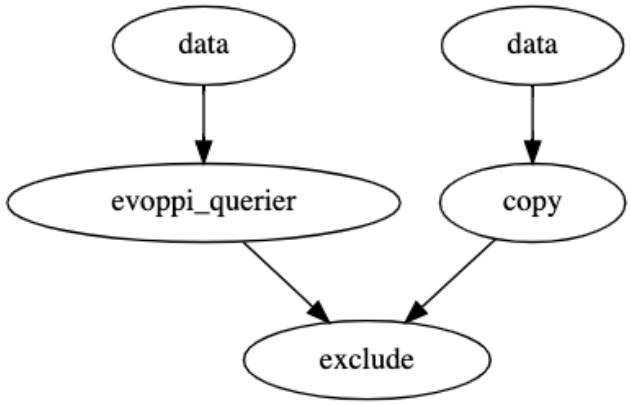

FIGURE 13. Graphical representation of the analysis performed on protocol_2.

\

3.6.3. Protocol 3

How different are the inferred structures from the AF and D-I-TASSER databases from the structures in the PDB database? In this protocol, TM scores are calculated between the structures retrieved from the AF and D-I-TASSER databases, and if they have a TM score higher than the one specified by the user, one of them is omitted (according to the user's specifications in priority order (in the case presented: D-I-TASSER and then AF). Then, PDB structures for the same proteins are retrieved from the PDB database and compared with the remaining structures, and if more than one structure is available for the same protein (because the AF and D-I-TASSER structures were more different than specified by the user), the one most similar to the one(s) available in the PDB database is selected. Figure 15 shows a graphical representation of the analysis.

Modules used: evoppi_querier, geneid2uniprotkb, getalphafoldpdb, getditasserpdb, tm-align-cutoff, getpdb, tm-align-pdb (Figure 14)

Execution time: 5 hours, 53 minutes and 8 seconds

Open | pipeline ~/Documents/jorge/docker-jorge/protocol_3 | Save

```
1 evoppi_querier input1 output1
2 geneid2uniprotkb output1 output2
3 getalphafoldpdb output2 output3
4 getditasserpdb output2 output3
5 tm-align-cutoff output3 output4
6 getpdb output2 output4
7 tm-align-pdb output4 output5
```

FIGURE 14. Pipeline file of protocol_3.

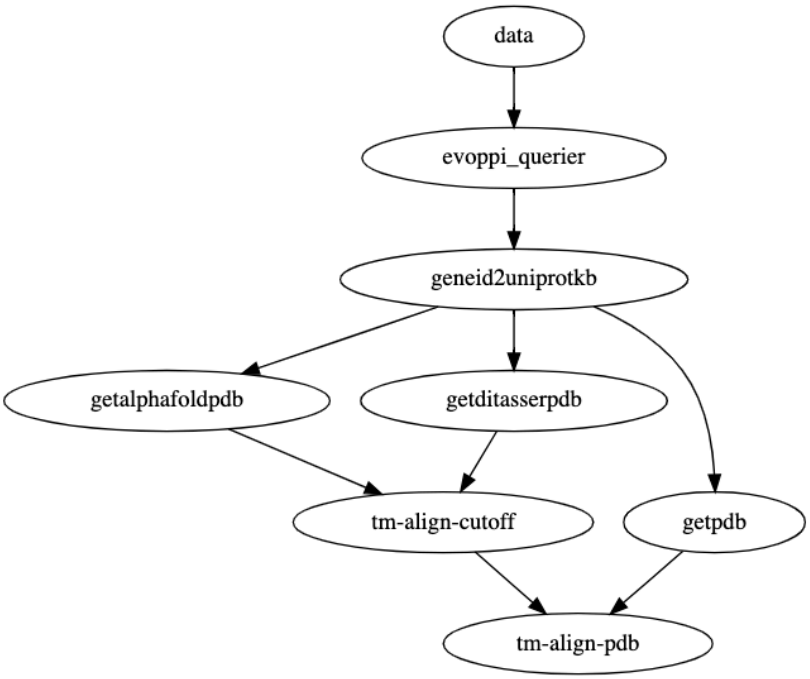

FIGURE 15. Graphical representation of the analysis performed on protocol_3.

3.6.4. Protocol 4

Full run, is suitable for retrieving data from the EvoPPI database, randomly selecting up to two interactors (as specified by the user), converting GeneIDs to UniProtKb numbers, filtering the result using protein distribution data from the Human Protein Atlas, retrieving the corresponding structures from the Alphafold database, using hdock as the docking program, PDBePISA to infer interface patterns and display them on PDB structures. Figure 17 shows a graphical representation of the analysis.

Modules used: evoppi_querier, geneid2uniprotkb, getalphafoldpdb, h_prot_atlas, getditasserpdb, tm-align-cutoff, getpdb, tm-align-pdb, copy, hdock, pisa_server_extract, tabulate, get_pattern and highlight_regions (Figure 16)

Execution time: 1 hour, 27 minutes and 47 seconds

Open | pipeline ~/Documents/jorge/docker-jorge/protocol_4 | Save

```
1 evoppi_querier input1 output1
2 geneid2uniprotkb output1 output2
3 h_prot_atlas output2 output3
4 getalphafoldpdb output3 output4
5 getpdb output3 output4
6 tm-align-pdb output4 output5
7 copy output5 PDBs/Ligands
8 getalphafoldpdb input2 output6
9 getpdb input2 output6
10 tm-align-pdb output6 output7
11 copy output7 PDBs/Receptor
12 hdock PDBs/Ligands output8
13 pisa_server_extract output8 output9
14 tabulate output9 output10
15 get_pattern output10 output11
16 highlight_regions output11 output12
```

FIGURE 16. Pipeline file of protocol_4.

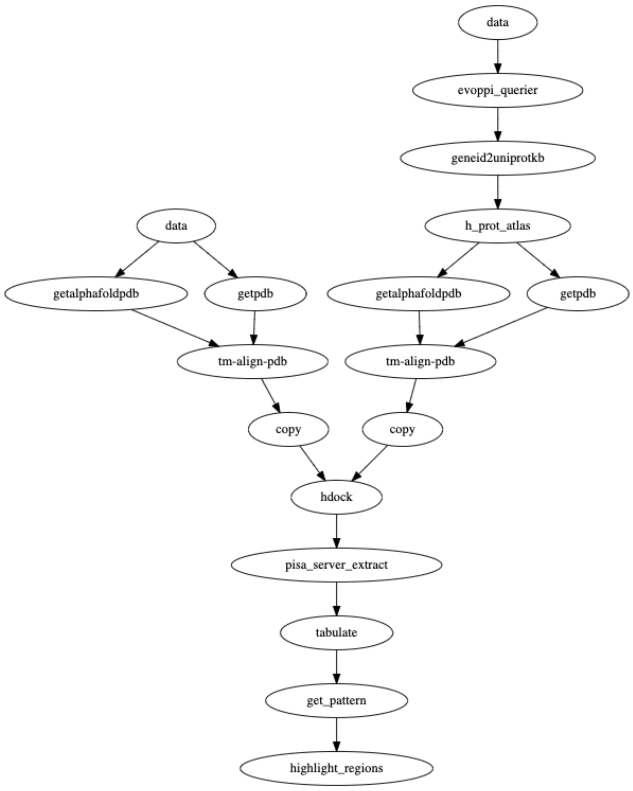

FIGURE 17. Graphical representation of the analysis performed on protocol_4.

4. Discussion

The auto-p2docking pipeline has demonstrated significant potential in providing a flexible and robust framework for PPI analysis. However, several improvements can further enhance its usability, efficiency and impact. Here I discuss key proposals that focus on improving usability, support, and advanced capabilities.

One of the most important improvements is the development of a graphical user interface (GUI). A user-friendly GUI would greatly simplify the configuration and execution of the pipeline, making it more accessible to users with varying levels of technical expertise. This visual approach could streamline the setup process, reducing the learning curve and allowing researchers to focus on their analyses rather than technical details. Enhancing the error checking feature to provide detailed diagnostic messages and potential solutions for common problems can significantly improve the process. By providing clear and actionable feedback, users can quickly identify and resolve problems, leading to smoother and more reliable pipeline execution. Establishing an online community forum or support platform can encourage collaboration, knowledge sharing and continuous improvement. A robust support system where users can share experiences, ask questions and provide feedback would improve the overall user experience. Developing interactive tutorials can help new users quickly become familiar with the pipeline's capabilities and best practices. These resources would serve as valuable tools for training, ensuring that users can effectively use the pipeline right from the start. Exploring the integration of machine learning algorithms into the pipeline can open new ways of data analysis, pattern recognition and predictive modelling. Machine learning can enhance the analytical capabilities of auto-p2docking, allowing researchers to uncover deeper insights and make more informed decisions based on their data. Another key improvement is the ability to easily include more modules from other programs. The open structure of the pipeline is designed to facilitate the inclusion of additional modules, even from software that does not yet exist. This flexibility ensures that the pipeline can continuously evolve and incorporate the latest advancements in PPI analysis.

By implementing these enhancements, auto-p2docking can become a more powerful, user-friendly and versatile tool for protein research. A user-friendly interface, advanced error handling, strong user community support, interactive tutorials, machine learning integration and the ability to easily include new modules are key areas that can significantly improve the functionality and user experience of the pipeline. These improvements will ultimately drive the use of auto-p2docking for the analysis of PPIs.

5. Conclusion

The auto-P2 docking pipeline represents a significant advance in the field of PPI analysis, addressing and overcoming many of the limitations of traditional experimental methods. It is not intended to replace experimental methods such as nuclear magnetic resonance (NMR) or yeast two-hybrid (Y2H) assays, but rather to complement them by providing a more efficient and time-saving approach. By performing preliminary *in silico* analyses, researchers can prioritize interactions for laboratory validation, reducing the number of experiments and focusing resources on the most promising candidates.

One of the most notable features of auto-p2docking is its modular design, which includes 22 individual modules that can be organized and combined according to the user's specific needs. This flexibility allows the creation of numerous pipeline configurations to suit different research requirements. Key features such as intermediate file storage for traceability, automated error checking, version control, comprehensive user documentation and graphical visualization of the pipeline further enhance its usability. Together, these features contribute to a more efficient workflow that significantly reduces analysis time and minimizes the risk of error. The ability to automate traditionally manual and error-prone processes is a major benefit, allowing researchers to focus on data interpretation and scientific inquiry rather than the mechanics of analysis. Auto-p2docking's ability to deliver faster and more accurate results makes it an invaluable tool for the computational biology community. The inclusion of predefined protocols within the auto-p2docking pipeline further enhances its utility by providing structured and tested workflows for different types of protein analysis.

In addition, auto-p2docking promotes open science and collaborative research by making bioinformatics tools available to a wider audience. Its design allows researchers with limited computational expertise to effectively use advanced methods for PPI analysis. The modular architecture, combined with Docker's reproducibility, makes auto-p2docking a powerful tool to meet the diverse needs of the scientific community and drive advances in our understanding of protein interactions and their impact on health and disease.

6. References

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., & Walter, P. (2002). *Molecular Biology of The Cell*.

2. Phizicky, E. M., & Fields, S. (1995). Protein-protein interactions: Methods for detection and analysis. *Microbiol Rev*, *59*, 94–123.

3. Gouveia, P. S. (2021). *A Docker-oriented Pipeline for the analysis of Protein- Protein Interactions*.

4. Nooren, I. M., & Thornton, J. M. (2003). Diversity of protein-protein interactions. *Embo j*, *22*, 3486–3492.

5. De Las Rivas, J., & Fontanillo, C. (2010). Protein–Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. *PLOS Computational Biology*, *6*, e1000807.

6. Cohen, M., Reichmann, D., Neuvirth, H., & Schreiber, G. (2008). Similar chemistry, but different bond preferences in inter versus intra-protein interactions. *Proteins: Structure, Function, and Bioinformatics*, *72*, 741–753.

7. Bahadur, R. P., Chakrabarti, P., Rodier, F., & Janin, J. (2004). A dissection of specific and non-specific protein-protein interfaces. *J Mol Biol*, *336*, 943–955.

8. Tripathi, A., & Varadarajan, R. (2014). Residue specific contributions to stability and activity inferred from saturation mutagenesis and deep sequencing. *Curr Opin Struct Biol*, *24*, 63–71.

9. Braun, P., & Gingras, A.-C. (2012). History of protein-protein interactions: From egg-white to complex networks. *Proteomics*, *12*(10), 1478–1498. https://doi.org/10.1002/pmic.201100563

10. Murakami, Y., Tripathi, L. P., Prathipati, P., & Mizuguchi, K. (2017). Network analysis and in silico prediction of protein-protein interactions with applications in drug discovery. *Current Opinion in Structural Biology*, *44*, 134–142. https://doi.org/10.1016/j.sbi.2017.02.005

11. Wells, J. A., & McClendon, C. L. (2007). Reaching for high-hanging fruit in drug discovery at protein–protein interfaces. *Nature*, *450*, 1001–1009.

12. Jubb, H., Blundell, T. L., & Ascher, D. B. (2015). Flexibility and small pockets at protein–protein interfaces: New insights into druggability. *Progress in Biophysics and Molecular Biology*, *119*, 2–9.

13. Prathipati, P., & Mizuguchi, K. (2016). Systems Biology Approaches to a Rational Drug Discovery Paradigm. *Current Topics in Medicinal Chemistry*, *16*(9), 1009–1025. https://doi.org/10.2174/1568026615666150826114524

14. Tuncbag, N., Kar, G., Keskin, O., Gursoy, A., & Nussinov, R. (2009). A survey of available tools and web servers for analysis of protein-protein interactions and interfaces. *Briefings in Bioinformatics*, *10*(3), 217–232. https://doi.org/10.1093/bib/bbp001

15. Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., & Rothberg, J. M. (2000). A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, *403*(6770), 623–627. https://doi.org/10.1038/35001009

16. Zhou, J., Thompson, D. K., Xu, Y., & Tiedje, J. M. (2004). *Microbial Functional Genomics*. John Wiley & Sons.

17. Salwinski, L., & Eisenberg, D. (2003). Computational methods of analysis of protein–protein interactions. *Current Opinion in Structural Biology*, *13*(3), 377–382. https://doi.org/10.1016/S0959-440X(03)00070-8

18. Blikstad, C., & Ivarsson, Y. (2015). High-throughput methods for identification of protein-protein interactions involving short linear motifs. *Cell Communication and Signaling*, *13*(1), 38. https://doi.org/10.1186/s12964-015-0116-8

19. Ho, Y., Gruhler, A., Heilbut, A., Bader, G. D., Moore, L., Adams, S.-L., Millar, A., Taylor, P., Bennett, K., Boutilier, K., Yang, L., Wolting, C., Donaldson, I., Schandorff, S.,

Shewnarane, J., Vo, M., Taggart, J., Goudreault, M., Muskat, B., … Tyers, M. (2002). Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. *Nature*, *415*(6868), 180–183. https://doi.org/10.1038/415180a

20. Zhu, H., Bilgin, M., Bangham, R., Hall, D., Casamayor, A., Bertone, P., Lan, N., Jansen, R., Bidlingmaier, S., Houfek, T., Mitchell, T., Miller, P., Dean, R. A., Gerstein, M., & Snyder, M. (2001). Global Analysis of Protein Activities Using Proteome Chips. *Science*, *293*(5537), 2101–2105. https://doi.org/10.1126/science.1062191

21. Tong, A. H. Y., Drees, B., Nardelli, G., Bader, G. D., Brannetti, B., Castagnoli, L., Evangelista, M., Ferracuti, S., Nelson, B., Paoluzi, S., Quondam, M., Zucconi, A., Hogue, C. W. V., Fields, S., Boone, C., & Cesareni, G. (2002). A Combined Experimental and Computational Strategy to Define Protein Interaction Networks for Peptide Recognition Modules. *Science*, *295*(5553), 321–324. https://doi.org/10.1126/science.1064987

22. Smyth, M. S. (2000). X Ray crystallography. *Molecular Pathology*, *53*(1), 8–14. https://doi.org/10.1136/mp.53.1.8

23. Brünger, A. T., Adams, P. D., Clore, G. M., DeLano, W. L., Gros, P., Grosse-Kunstleve, R. W., Jiang, J. S., Kuszewski, J., Nilges, M., Pannu, N. S., Read, R. J., Rice, L. M., Simonson, T., & Warren, G. L. (1998). Crystallography & NMR System: A New Software Suite for Macromolecular Structure Determination. *Acta Crystallographica Section D Biological Crystallography*, *54*(5), 905–921. https://doi.org/10.1107/S0907444998003254

24. Bonvin, A. M., Boelens, R., & Kaptein, R. (2005). NMR analysis of protein interactions. *Curr Opin Chem Biol*, *9*, 501–508.

25. Lua, R. C., Marciano, D. C., Katsonis, P., Adikesavan, A. K., Wilkins, A. D., & Lichtarge, O. (2014). Prediction and redesign of protein–protein interactions. *Prog Biophys Mol Biol*, *116*, 194–202.

26. Acuner Ozbabacan, S. E., Engin, H. B., Gursoy, A., & Keskin, O. (2011). Transient protein–protein interactions. *Protein Eng Des Sel*, *24*, 635–648.

27. Seet, B. T., Dikic, I., Zhou, M.-M., & Pawson, T. (2006). Reading protein modifications with interaction domains. *Nat Rev Mol Cell Biol*, *7*, 473–483.

28. Meszaros, B., Simon, I., & Dosztanyi, Z. (2009). Prediction of protein binding regions in disordered proteins. *PLoS Comput Biol*, *5*, e1000376.

29. Duan, G., & Walther, D. (2015). The roles of post-translational modifications in the context of protein interaction networks. *PLoS Comput Biol*, *11*, e1004049.

30. Babu, M. M., Kriwacki, R. W., & Pappu, R. V. (2012). Structural biology. Versatility from protein disorder. *Science*, *337*, 1460–1461.

31. Lin, T.-W., Wu, J.-W., & Chang, D. T.-H. (2013). Combining Phylogenetic Profiling-Based and Machine Learning-Based Techniques to Predict Functional Related Proteins. *PLoS One*, *8*, e75940.

32. Rao, V. S., Srinivas, K., Sujini, G. N., & Kumar, G. N. S. (2014). Protein-Protein Interaction Detection: Methods and Analysis. *International Journal of Proteomics*, *2014*, 1–12. https://doi.org/10.1155/2014/147648

33. Piehler, J. (2005). New methodologies for measuring protein interactions in vivo and in vitro. *Curr Opin Struct Biol*, *15*, 4–14.

34. Ngounou Wetie, A. G., Sokolowska, I., Woods, A. G., Roy, U., Loo, J. A., & Darie, C. C. (2013). Investigation of stable and transient protein-protein interactions: Past, present, and future. *Proteomics*, *13*, 538–557.

35. Huang, H., & Bader, J. S. (2009). Precision and recall estimates for two-hybrid screens. *Bioinformatics*, *25*, 372–378.

36. Gingras, A. C., Gstaiger, M., Raught, B., & Aebersold, R. (2007). Analysis of protein complexes using mass spectrometry. *Nat Rev Mol Cell Biol*, *8*, 645–654.

37. Serebriiskii, I. G., & Golemis, E. A. (2001). Two-hybrid system and false positives. Approaches to detection and elimination. *Methods Mol Biol*, *177*, 123–134.

38. Berggård, T., Linse, S., & James, P. (2007). Methods for the detection and analysis of protein–protein interactions. *Proteomics*, *7*, 2833–2842.

39. Tang, T., Zhang, X., Liu, Y., Peng, H., Zheng, B., Yin, Y., & Zeng, X. (2023).

Machine learning on protein–protein interaction prediction: Models, challenges and trends. *Briefings in Bioinformatics*, *24*(2), bbad076. https://doi.org/10.1093/bib/bbad076

40. Jessulat, M., Pitre, S., Gui, Y., Hooshyar, M., Omidi, K., Samanfar, B., Tan, L. H., Alamgir, M., Green, J., Dehne, F., & Golshani, A. (2011). Recent advances in protein–protein interaction prediction: Experimental and computational methods. *Expert Opinion on Drug Discovery*, *6*(9), 921–935. https://doi.org/10.1517/17460441.2011.603722

41. Gong, Y., Li, R., Fu, B., Liu, Y., Wang, J., Li, R., & Chen, D. Z. (2023). A CNN-LSTM Ensemble Model for Predicting Protein-Protein Interaction Binding Sites. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *20*(6), 3588–3599. https://doi.org/10.1109/TCBB.2023.3306948

42. Li, Y., Golding, G. B., & Ilie, L. (2021). DELPHI: Accurate deep ensemble model for protein interaction sites prediction. *Bioinformatics*, *37*(7), 896–904. https://doi.org/10.1093/bioinformatics/btaa750

43. Pinzi, L., & Rastelli, G. (2019). Molecular Docking: Shifting Paradigms in Drug Discovery. *International Journal of Molecular Sciences*, *20*(18), 4331. https://doi.org/10.3390/ijms20184331

44. Townsend-Nicholson, A., Altwaijry, N., Potterton, A., Morao, I., & Heifetz, A. (2019). Computational prediction of GPCR oligomerization. *Current Opinion in Structural Biology*, *55*, 178–184. https://doi.org/10.1016/j.sbi.2019.04.005

45. Xu, D., Xu, Y., & Uberbacher, E. (2000). Computational Tools For Protein Modeling. *Current Protein & Peptide Science*, *1*(1), 1–21. https://doi.org/10.2174/1389203003381469

46. Szklarczyk, D., Kirsch, R., Koutrouli, M., Nastou, K., Mehryary, F., Hachilif, R., Gable, A. L., Fang, T., Doncheva, N. T., Pyysalo, S., Bork, P., Jensen, L. J., & von Mering, C. (2023). The STRING database in 2023: Protein–protein association networks and functional enrichment analyses for any sequenced genome of interest. *Nucleic Acids Research*, *51*(D1), D638–D646. https://doi.org/10.1093/nar/gkac1000

47. Benstead-Hume, G., Chen, X., Hopkins, S. R., Lane, K. A., Downs, J. A., & Pearl, F. M. G. (2019). Predicting synthetic lethal interactions using conserved patterns in protein interaction networks. *PLOS Computational Biology*, *15*(4), e1006888. https://doi.org/10.1371/journal.pcbi.1006888

48. Sousa, A., Rocha, S., Vieira, J., Reboiro-Jato, M., López-Fernández, H., & Vieira, C. P. (2023). On the identification of potential novel therapeutic targets for spinocerebellar ataxia type 1 (SCA1) neurodegenerative disease using EvoPPI3. *Journal of Integrative Bioinformatics*, *20*(2), 20220056. https://doi.org/10.1515/jib-2022-0056

49. von Vázquez, N., Rocha, S., López-Fernández, H., Torres, A., Camacho, R., Fdez-Riverola, F., Vieira, J., Vieira, C. P., & Reboiro-Jato, M. (2019). *EvoPPI: A Web Application to Compare Protein-Protein Interactions (PPIs) from Different Databases and Species* (pp. 149–156). Springer International Publishing.

50. Vakser, I. A. (sem data). Protein-Protein Docking: From Interaction to Interactome. *Biophysical Journal*.

51. Siebenmorgen, T., & Zacharias, M. (2020). Computational prediction of protein–protein binding affinities. *WIREs Computational Molecular Science*, *10*(3), e1448. https://doi.org/10.1002/wcms.1448

52. Chen, R., Li, L., & Weng, Z. (2003). ZDOCK: An initial-stage protein-docking algorithm. *Proteins*, *52*(1), 80–87. https://doi.org/10.1002/prot.10389

53. Singh, A., Copeland, M. M., Kundrotas, P. J., & Vakser, I. A. (2024). GRAMM Web Server for Protein Docking. Em M. Gore & U. B. Jagtap (Eds.), *Computational Drug Discovery and Design* (Vol. 2714, pp. 101–112). Springer US. https://doi.org/10.1007/978-1-0716-3441-7_5

54. Mandell, J. G., Roberts, V. A., Pique, M. E., Kotlovyi, V., Mitchell, J. C., Nelson, E., Tsigelny, I., & Ten Eyck, L. F. (2001). Protein docking using continuum electrostatics and geometric fit. *Protein Engineering, Design and Selection*, *14*(2), 105–113. https://doi.org/10.1093/protein/14.2.105

55. Camacho, C. J., & Vajda, S. (2001). Protein docking along smooth association

pathways. *Proceedings of the National Academy of Sciences*, *98*(19), 10636–10641. https://doi.org/10.1073/pnas.181147798

56. Kozakov, D., Hall, D. R., Xia, B., Porter, K. A., Padhorny, D., Yueh, C., Beglov, D., & Vajda, S. (2017). The ClusPro web server for protein-protein docking. *Nature Protocols*, *12*(2), 255–278. https://doi.org/10.1038/nprot.2016.169

57. Comeau, S. R., Gatchell, D. W., Vajda, S., & Camacho, C. J. (2004). ClusPro: A fully automated algorithm for protein-protein docking. *Nucleic Acids Research*, *32*(Web Server issue), W96-99. https://doi.org/10.1093/nar/gkh354

58. Redington, P. K. (1992). MOLFIT: A computer program for molecular superposition. *Computers & Chemistry*, *16*(3), 217–222. https://doi.org/10.1016/0097-8485(92)80005-K

59. Gabb, H. A., Jackson, R. M., & Sternberg, M. J. E. (1997). Modelling protein docking using shape complementarity, electrostatics and biochemical information 1 1Edited by J. Thornton. *Journal of Molecular Biology*, *272*(1), 106–120. https://doi.org/10.1006/jmbi.1997.1203

60. J, M., Dw, G., Jc, P., & S, V. (2003). Combination of scoring functions improves discrimination in protein-protein docking. *Proteins*, *53*(4). https://doi.org/10.1002/prot.10473

61. Rm, J., Ha, G., & Mj, S. (1998). Rapid refinement of protein interfaces incorporating solvation: Application to the docking problem. *Journal of Molecular Biology*, *276*(1). https://doi.org/10.1006/jmbi.1997.1519

62. Kozakov, D., Brenke, R., Comeau, S. R., & Vajda, S. (2006). PIPER: An FFT-based protein docking program with pairwise potentials. *Proteins*, *65*(2), 392–406. https://doi.org/10.1002/prot.21117

63. Cheng, T. M.-K., Blundell, T. L., & Fernandez-Recio, J. (2007). pyDock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking. *Proteins: Structure, Function, and Bioinformatics*, *68*(2), 503–515. https://doi.org/10.1002/prot.21419

64. Yan, Y., Wen, Z., Wang, X., & Huang, S. (2017). Addressing recent docking challenges: A hybrid strategy to integrate template-based and free protein-protein docking. *Proteins: Structure, Function, and Bioinformatics*, *85*(3), 497–512. https://doi.org/10.1002/prot.25234

65. Yan, Y., Tao, H., He, J., & Huang, S.-Y. (2020). The HDOCK server for integrated protein–protein docking. *Nature Protocols*, *15*(5), 1829–1852. https://doi.org/10.1038/s41596-020-0312-x

66. Ghoorah, A. W., Devignes, M., Smaïl-Tabbone, M., & Ritchie, D. W. (2013). Protein docking using case-based reasoning. *Proteins: Structure, Function, and Bioinformatics*, *81*(12), 2150–2158. https://doi.org/10.1002/prot.24433

67. Garzon, J. I., Lopéz-Blanco, J. R., Pons, C., Kovacs, J., Abagyan, R., Fernandez-Recio, J., & Chacon, P. (2009). FRODOCK: A new approach for fast rotational protein–protein docking. *Bioinformatics*, *25*(19), 2544. https://doi.org/10.1093/bioinformatics/btp447

68. J, Y., M, V., J, A., J, R., P, T., & R, G. (2016). InterEvDock: A docking server to predict the structure of protein-protein interactions using evolutionary information. *Nucleic Acids Research*, *44*(W1). https://doi.org/10.1093/nar/gkw340

69. Huang, S.-Y., & Zou, X. (2010). MDockPP: A hierarchical approach for protein-protein docking and its application to CAPRI rounds 15–19. *Proteins*, *78*(15), 3096. https://doi.org/10.1002/prot.22797

70. Kong, R., Wang, F., Zhang, J., Wang, F., & Chang, S. (2019). CoDockPP: A Multistage Approach for Global and Site-Specific Protein–Protein Docking. *Journal of Chemical Information and Modeling*, *59*(8), 3556–3564. https://doi.org/10.1021/acs.jcim.9b00445

71. Yan, Y., Tao, H., & Huang, S.-Y. (2018). HSYMDOCK: A docking web server for predicting the structure of protein homo-oligomers with Cn or Dn symmetry. *Nucleic Acids Research*, *46*(Web Server issue), W423. https://doi.org/10.1093/nar/gky398

72. Ritchie, D. W., & Grudinin, S. (2016). Spherical polar Fourier assembly of protein complexes with arbitrary point group symmetry. *Journal of Applied Crystallography*, *49*(1), 158–167. https://doi.org/10.1107/S1600576715022931

73. Ds, G., & Aj, O. (1990). Automated docking of substrates to proteins by simulated annealing. *Proteins*, *8*(3). https://doi.org/10.1002/prot.340080302

74. Js, T., & Rm, B. (2000). DARWIN: A program for docking flexible molecules. *Proteins*, *41*(2). https://pubmed.ncbi.nlm.nih.gov/10966571/

75. J, E.-R., Yd, Y., & D, K. (2012). Multi-LZerD: Multiple protein docking for asymmetric complexes. *Proteins*, *80*(7). https://doi.org/10.1002/prot.24079

76. G, J., P, W., Rc, G., Ar, L., & R, T. (1997). Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, *267*(3). https://doi.org/10.1006/jmbi.1996.0897

77. Lyskov, S., & Gray, J. J. (2008). The RosettaDock server for local protein-protein docking. *Nucleic acids research*, *36*, W233–W238.

78. Neves, M. A. C., Totrov, M., & Abagyan, R. (2012). Docking and scoring with ICM: The benchmarking results and strategies for improvement. *Journal of Computer-Aided Molecular Design*, *26*(6), 675. https://doi.org/10.1007/s10822-012-9547-0

79. Dominguez, C., Boelens, R., & Bonvin, A. M. J. J. (2003). HADDOCK: A protein-protein docking approach based on biochemical or biophysical information. *Journal of the American Chemical Society*, *125*(7), 1731–1737. https://doi.org/10.1021/ja026939x

80. Sun, D., Liu, S., & Gong, X. (2020). Review of multimer protein–protein interaction complex topology and structure prediction*. *Chinese Physics B*, *29*(10), 108707. https://doi.org/10.1088/1674-1056/abb659

81. Savojardo, C., Martelli, P. L., & Casadio, R. (2020). Protein–Protein Interaction Methods and Protein Phase Separation. *Annual Review of Biomedical Data Science*, *3*(Volume 3, 2020), 89–112. https://doi.org/10.1146/annurev-biodatasci-011720-104428

82. Zhang, J., & Kurgan, L. (2019). SCRIBER: Accurate and partner type-specific prediction of protein-binding residues from proteins sequences. *Bioinformatics*, *35*(14), i343–i353. https://doi.org/10.1093/bioinformatics/btz324

83. Murakami, Y., & Mizuguchi, K. (2010). Applying the Naïve Bayes classifier with kernel density estimation to the prediction of protein–protein interaction sites. *Bioinformatics*, *26*(15), 1841–1848. https://doi.org/10.1093/bioinformatics/btq302

84. de Vries, S. J., & Bonvin, A. M. J. J. (2011). CPORT: A consensus interface predictor and its performance in prediction-driven docking with HADDOCK. *PloS One*, *6*(3), e17695. https://doi.org/10.1371/journal.pone.0017695

85. Krissinel, E., & Henrick, K. (2007). Inference of Macromolecular Assemblies from Crystalline State. *Journal of Molecular Biology*, *372*, 774–797.

86. Jm, S., & T, H.-G. (2000). Matching simulation and experiment: A new simplified model for simulating protein folding. *Journal of Computational Biology : A Journal of Computational Molecular Cell Biology*, *7*(3–4). https://doi.org/10.1089/106652700750050899

87. Yang, J., & Zhang, Y. (2015). I-TASSER server: New development for protein structure and function predictions. *Nucleic Acids Research*, *43*(W1), W174–W181. https://doi.org/10.1093/nar/gkv342

88. Dt, J. (1997). Successful ab initio prediction of the tertiary structure of NK-lysin using multiple sequences and recognized supersecondary structural motifs. *Proteins*, *Suppl 1*. https://doi.org/10.1002/(sici)1097-0134(1997)1+<185::aid-prot24>3.3.co;2-t

89. Göbel, U., Sander, C., Schneider, R., & Valencia, A. (1994). Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Bioinformatics*, *18*(4), 309–317. https://doi.org/10.1002/prot.340180402

90. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

91. Elofsson, A. (2023). Progress at protein structure prediction, as seen in CASP15. *Current Opinion in Structural Biology*, *80*, 102594. https://doi.org/10.1016/j.sbi.2023.102594

92. Jänes, J., & Beltrao, P. (2024). Deep learning for protein structure prediction and

design—Progress and applications. *Molecular Systems Biology*, *20*(3), 162–169. https://doi.org/10.1038/s44320-024-00016-x

93. Scardino, V., Di Filippo, J. I., & Cavasotto, C. N. (2023). How good are AlphaFold models for docking-based virtual screening? *iScience*, *26*(1), 105920. https://doi.org/10.1016/j.isci.2022.105920

94. Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., & Tasumi, M. (1977). The protein data bank: A computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, *112*(3), 535–542. https://doi.org/10.1016/S0022-2836(77)80200-3

95. Zheng, W., Wuyun, Q., Freddolino, P. L., & Zhang, Y. (2023). Integrating deep learning, threading alignments, and a MULTI-MSA strategy for high-quality protein monomer and complex structure prediction in CASP15. *Proteins: Structure, Function, and Bioinformatics*, *91*(12), 1684–1703. https://doi.org/10.1002/prot.26585

96. Zhang, Y., & Skolnick, J. (2005). TM-align: A protein structure alignment algorithm based on the TM-score. *Nucleic Acids Res*, *33*, 2302–2309.

97. Potterton, L., Agirre, J., Ballard, C., Cowtan, K., Dodson, E., Evans, P. R., Jenkins, H. T., Keegan, R., Krissinel, E., Stevenson, K., Lebedev, A., McNicholas, S. J., Nicholls, R. A., Noble, M., Pannu, N. S., Roth, C., Sheldrick, G., Skubak, P., Turkenburg, J., … Wojdyr, M. (2018). *CCP* 4 *i* 2: The new graphical user interface to the *CCP* 4 program suite. *Acta Crystallographica Section D Structural Biology*, *74*(2), 68–84. https://doi.org/10.1107/S2059798317016035

98. Vázquez, N., Rocha, S., López-Fernández, H., Torres, A., Camacho, R., Fdez-Riverola, F., Vieira, J., Vieira, C. P., & Reboiro-Jato, M. (2019). EvoPPI 1.0: A Web Platform for Within- and Between-Species Multiple Interactome Comparisons and Application to Nine PolyQ Proteins Determining Neurodegenerative Diseases. *Interdisciplinary Sciences: Computational Life Sciences*, *11*(1), 45–56. https://doi.org/10.1007/s12539-019-00317-y

99. Reboiro-Jato, M., Vieira, J., Rocha, S., Sousa, A. D., López-Fernández, H., & Vieira, C. P. (2023). EvoPPI 2: A Web and Local Platform for the Comparison of Protein–Protein Interaction Data from Multiple Sources from the Same and Distinct Species. Em F. Fdez-Riverola, M. Rocha, M. S. Mohamad, S. Caraiman, & A. B. Gil-González (Eds.), *Practical Applications of Computational Biology and Bioinformatics, 16th International Conference (PACBB 2022)* (Vol. 553, pp. 101–110). Springer International Publishing. https://doi.org/10.1007/978-3-031-17024-9_10

100. Stark, C. (2006). BioGRID: A general repository for interaction datasets. *Nucleic Acids Research*, *34*(90001), D535–D539. https://doi.org/10.1093/nar/gkj109

101. Oughtred, R., Stark, C., Breitkreutz, B.-J., Rust, J., Boucher, L., Chang, C., Kolas, N., O'Donnell, L., Leung, G., McAdam, R., Zhang, F., Dolma, S., Willems, A., Coulombe-Huntington, J., Chatr-aryamontri, A., Dolinski, K., & Tyers, M. (2019). The BioGRID interaction database: 2019 update. *Nucleic Acids Research*, *47*(D1), D529–D541. https://doi.org/10.1093/nar/gky1079

102. Luck, K., Kim, D.-K., Lambourne, L., Spirohn, K., Begg, B. E., Bian, W., Brignall, R., Cafarelli, T., Campos-Laborie, F. J., Charloteaux, B., Choi, D., Coté, A. G., Daley, M., Deimling, S., Desbuleux, A., Dricot, A., Gebbia, M., Hardy, M. F., Kishore, N., … Calderwood, M. A. (2020). A reference map of the human binary protein interactome. *Nature*, *580*(7803), 402–408. https://doi.org/10.1038/s41586-020-2188-x

103. Murali, T., Pacifico, S., Yu, J., Guest, S., Roberts, G. G., & Finley, R. L. (2011). DroID 2011: A comprehensive, integrated resource for protein, transcription factor, RNA and gene interactions for Drosophila. *Nucleic Acids Research*, *39*(suppl_1), D736–D743. https://doi.org/10.1093/nar/gkq1092

104. Attrill, H., Falls, K., Goodman, J. L., Millburn, G. H., Antonazzo, G., Rey, A. J., Marygold, S. J., & the FlyBase consortium. (2016). FlyBase: Establishing a Gene Group resource for *Drosophila melanogaster*. *Nucleic Acids Research*, *44*(D1), D786–D792. https://doi.org/10.1093/nar/gkv1046

105. Alanis-Lobato, G., Andrade-Navarro, M. A., & Schaefer, M. H. (2017). HIPPIE

v2.0: Enhancing meaningfulness and reliability of protein–protein interaction networks. *Nucleic Acids Research*, *45*(D1), D408–D414. https://doi.org/10.1093/nar/gkw985

106. López, Y., Nakai, K., & Patil, A. (2015). HitPredict version 4: Comprehensive reliability scoring of physical protein–protein interactions from more than 100 species. *Database*, *2015*, bav117. https://doi.org/10.1093/database/bav117

107. Persico, M., Ceol, A., Gavrila, C., Hoffmann, R., Florio, A., & Cesareni, G. (2005). HomoMINT: An inferred human network based on orthology mapping of protein interactions discovered in model organisms. *BMC Bioinformatics*, *6*(S4), S21. https://doi.org/10.1186/1471-2105-6-S4-S21

108. Meyer, M. J., Das, J., Wang, X., & Yu, H. (2013). INstruct: A database of high-quality 3D structurally resolved protein interactome networks. *Bioinformatics*, *29*(12), 1577–1579. https://doi.org/10.1093/bioinformatics/btt181

109. Mosca, R., Céol, A., & Aloy, P. (2013). Interactome3D: Adding structural details to protein networks. *Nature Methods*, *10*(1), 47–53. https://doi.org/10.1038/nmeth.2289

110. Calderone, A., Castagnoli, L., & Cesareni, G. (2013). mentha: A resource for browsing integrated protein-interaction networks. *Nature Methods*, *10*(8), 690–691. https://doi.org/10.1038/nmeth.2561

111. Zanzoni, A., Montecchi-Palazzi, L., Quondam, M., Ausiello, G., Helmer-Citterich, M., & Cesareni, G. (2002). MINT: A Molecular INTeraction database. *FEBS Letters*, *513*(1), 135–140. https://doi.org/10.1016/S0014-5793(01)03293-8

112. Licata, L., Briganti, L., Peluso, D., Perfetto, L., Iannuccelli, M., Galeota, E., Sacco, F., Palma, A., Nardozza, A. P., Santonico, E., Castagnoli, L., & Cesareni, G. (2012). MINT, the molecular interaction database: 2012 update. *Nucleic Acids Research*, *40*(D1), D857–D861. https://doi.org/10.1093/nar/gkr930

113. Cowley, M. J., Pinese, M., Kassahn, K. S., Waddell, N., Pearson, J. V., Grimmond, S. M., Biankin, A. V., Hautaniemi, S., & Wu, J. (2012). PINA v2.0: Mining interactome modules. *Nucleic Acids Research*, *40*(D1), D862–D865. https://doi.org/10.1093/nar/gkr967

114. Pagel, P., Mewes, H.-W., & Frishman, D. (2004). Conservation of protein–protein interactions – lessons from ascomycota. *Trends in Genetics*, *20*(2), 72–76. https://doi.org/10.1016/j.tig.2003.12.007

115. Thul, P. J., & Lindskog, C. (2018). The human protein atlas: A spatial map of the human proteome. *Protein Science*, *27*(1), 233–244. https://doi.org/10.1002/pro.3307

116. Digre, A., & Lindskog, C. (2021). The Human Protein Atlas—Spatial localization of the human proteome in health and disease. *Protein Science*, *30*(1), 218–233. https://doi.org/10.1002/pro.3987

117. wwPDB consortium, Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Costanzo, L. D., Christie, C., Duarte, J. M., Dutta, S., Feng, Z., Ghosh, S., Goodsell, D. S., Green, R. K., Guranovic, V., Guzenko, D., Hudson, B. P., Liang, Y., Lowe, R., … Ioannidis, Y. E. (2019). Protein Data Bank: The single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, *47*(D1), D520–D528. https://doi.org/10.1093/nar/gky949

118. Mitchell, A. L., Almeida, A., Beracochea, M., Boland, M., Burgin, J., Cochrane, G., Crusoe, M. R., Kale, V., Potter, S. C., Richardson, L. J., Sakharova, E., Scheremetjew, M., Korobeynikov, A., Shlemov, A., Kunyavskaya, O., Lapidus, A., & Finn, R. D. (2019). MGnify: The microbiome analysis resource in 2020. *Nucleic Acids Research*, gkz1035. https://doi.org/10.1093/nar/gkz1035

119. Steinegger, M., Mirdita, M., & Söding, J. (2019). Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, *16*(7), 603–606. https://doi.org/10.1038/s41592-019-0437-4

120. Peng, C.-X., Liang, F., Xia, Y.-H., Zhao, K.-L., Hou, M.-H., & Zhang, G.-J. (2024). Recent Advances and Challenges in Protein Structure Prediction. *Journal of Chemical Information and Modeling*, *64*(1), 76–95. https://doi.org/10.1021/acs.jcim.3c01324

121. Kryshtafovych, A., Schwede, T., Topf, M., Fidelis, K., & Moult, J. (2021). Critical assessment of methods of protein structure prediction (CASP)—Round XIV. *Proteins: Structure, Function, and Bioinformatics*, *89*(12), 1607–1617. https://doi.org/10.1002/prot.26237

122. Yang, Z., Zeng, X., Zhao, Y., & Chen, R. (2023). AlphaFold2 and its applications in the fields of biology and medicine. *Signal Transduction and Targeted Therapy*, *8*(1), 115. https://doi.org/10.1038/s41392-023-01381-z

123. Roy, A., Kucukural, A., & Zhang, Y. (2010). I-TASSER: A unified platform for automated protein structure and function prediction. *Nature Protocols*, *5*(4), 725–738. https://doi.org/10.1038/nprot.2010.5

124. Yang, J., Yan, R., Roy, A., Xu, D., Poisson, J., & Zhang, Y. (2015). The I-TASSER Suite: Protein structure and function prediction. *Nature Methods*, *12*(1), 7–8. https://doi.org/10.1038/nmeth.3213

125. Yang, J., Zhang, W., He, B., Walker, S. E., Zhang, H., Govindarajoo, B., Virtanen, J., Xue, Z., Shen, H. B., & Zhang, Y. (2016). Template-based protein structure prediction in CASP11 and retrospect of I-TASSER in the last decade. *Proteins*, *84 Suppl 1*, 233–246.

126. Peng, Z., Wang, W., Han, R., Zhang, F., & Yang, J. (2022). Protein structure prediction in the deep learning era. *Current Opinion in Structural Biology*, *77*, 102495. https://doi.org/10.1016/j.sbi.2022.102495

127. Kihara, D., Lu, H., Kolinski, A., & Skolnick, J. (2001). TOUCHSTONE: An *ab initio* protein structure prediction method that uses threading-based tertiary restraints. *Proceedings of the National Academy of Sciences*, *98*(18), 10125–10130. https://doi.org/10.1073/pnas.181328398

128. Zhang, C., Freddolino, P. L., & Zhang, Y. (2017). COFACTOR: Improved protein function prediction by combining structure, sequence and protein–protein interaction information. *Nucleic Acids Research*, *45*(W1), W291–W299. https://doi.org/10.1093/nar/gkx366

129. Zheng, W., Wuyun, Q., Zhou, X., Li, Y., Freddolino, P. L., & Zhang, Y. (2022). LOMETS3: Integrating deep learning and profile alignment for advanced protein template recognition and function annotation. *Nucleic Acids Research*, *50*(W1), W454–W464. https://doi.org/10.1093/nar/gkac248

130. Hubbard, T. J., Murzin, A. G., Brenner, S. E., & Chothia, C. (1997). SCOP: a structural classification of proteins database. *Nucleic acids research*, *25*, 236–239.

131. Moult, J., Fidelis, K., Zemla, A., & Hubbard, T. (2003). Critical assessment of methods of protein structure prediction (CASP)-round V. *Proteins*, *53*, 334–339.

132. Skolnick, J., Fetrow, J. S., & Kolinski, A. (2000). Structural genomics and its importance for gene function analysis. *Nat Biotechnol*, *18*, 283–287.

133. Baker, D., & Sali, A. (2001). Protein structure prediction and structural genomics. *Science*, *294*, 93–96.

134. Hasegawa, H., & Holm, L. (2009). Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, *19*(3), 341–348. https://doi.org/10.1016/j.sbi.2009.04.003

135. Zhang, Y., & Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. *Proteins*, *57*, 702–710.

136. Maheshwari, S., & Brylinski, M. (2015). Predicting protein interface residues using easily accessible on-line resources. *Briefings in Bioinformatics*, *16*(6), 1025–1034. https://doi.org/10.1093/bib/bbv009

137. Xue, L. C., Dobbs, D., Bonvin, A. M. J. J., & Honavar, V. (2015). Computational prediction of protein interfaces: A review of data driven methods. *FEBS Letters*, *589*(23), 3516–3526. https://doi.org/10.1016/j.febslet.2015.10.003

138. Zhou, H.-X., & Qin, S. (2007). Interaction-site prediction for protein complexes: A critical assessment. *Bioinformatics*, *23*(17), 2203–2209. https://doi.org/10.1093/bioinformatics/btm323

139. Porollo, A., & Meller, J. (2007). Prediction-Based Fingerprints of Protein – Protein Interactions. *Proteins-Structure Function and Bioinformatics*, *66*, 630–645.

140. Savojardo, C., Fariselli, P., Martelli, P. L., & Casadio, R. (2017). ISPRED4: Interaction sites PREDiction in protein structures with a refining grammar model. *Bioinformatics (Oxford, England)*, *33*(11), 1656–1663. https://doi.org/10.1093/bioinformatics/btx044

141. Andreani, J., Quignot, C., & Guerois, R. (2020). Structural prediction of protein

interactions and docking using conservation and coevolution. *WIREs Computational Molecular Science*, *10*(6), e1470. https://doi.org/10.1002/wcms.1470

142. Nooren, I. M., & Thornton, J. M. (2003). Structural characterisation and functional significance of transient protein-protein interactions. *J Mol Biol*, *325*, 991–1018.

143. Desta, I. T., Porter, K. A., Xia, B., Kozakov, D., & Vajda, S. (2020). Performance and Its Limits in Rigid Body Protein-Protein Docking. *Structure*, *28*(9), 1071-1081.e3. https://doi.org/10.1016/j.str.2020.06.006

144. Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., & Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, *28*(1), 235–242. https://doi.org/10.1093/nar/28.1.235

145. de Vries, S. J., van Dijk, M., & Bonvin, A. M. J. J. (2010). The HADDOCK web server for data-driven biomolecular docking. *Nature Protocols*, *5*(5), 883–897. https://doi.org/10.1038/nprot.2010.32

146. Hopf, T. A., Schärfe, C. P. I., Rodrigues, J. P. G. L. M., Green, A. G., Kohlbacher, O., Sander, C., Bonvin, A. M. J. J., & Marks, D. S. (2014). Sequence co-evolution gives 3D contacts and structures of protein complexes. *eLife*, *3*, e03430. https://doi.org/10.7554/eLife.03430

147. Vreven, T., Hwang, H., Pierce, B. G., & Weng, Z. (2014). Evaluating template-based and template-free protein-protein complex structure prediction. *Briefings in Bioinformatics*, *15*(2), 169–176. https://doi.org/10.1093/bib/bbt047

148. Porter, K. A., Desta, I., Kozakov, D., & Vajda, S. (2019). What method to use for protein–protein docking? *Current Opinion in Structural Biology*, *55*, 1–7. https://doi.org/10.1016/j.sbi.2018.12.010

149. Gray, J. J., Moughon, S., Wang, C., Schueler-Furman, O., Kuhlman, B., Rohl, C. A., & Baker, D. (2003). Protein–Protein Docking with Simultaneous Optimization of Rigid-body Displacement and Side-chain Conformations. *Journal of Molecular Biology*, *331*(1), 281–299. https://doi.org/10.1016/S0022-2836(03)00670-3

150. Huang, S.-Y. (2015). Exploring the potential of global protein–protein docking: An overview and critical assessment of current programs for automatic ab initio docking. *Drug Discovery Today*, *20*(8), 969–977. https://doi.org/10.1016/j.drudis.2015.03.007

151. Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A. A., Aflalo, C., & Vakser, I. A. (1992). Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, *89*(6), 2195. https://doi.org/10.1073/pnas.89.6.2195

152. Ritchie, D. W. (sem data). Recent Progress and Future Directions in Protein-Protein Docking. *Current Protein & Peptide Science*, *9*(1), 1–15.

153. Rosell, M., Rodríguez-Lumbreras, L. A., & Fernández-Recio, J. (2020). Modeling of Protein Complexes and Molecular Assemblies with pyDock. Em D. Kihara (Ed.), *Protein Structure Prediction* (pp. 175–198). Springer US. https://doi.org/10.1007/978-1-0716-0708-4_10

154. Case, D. A., Cheatham, T. E., Darden, T., Gohlke, H., Luo, R., Merz, K. M., Onufriev, A., Simmerling, C., Wang, B., & Woods, R. J. (2005). The Amber biomolecular simulation programs. *Journal of Computational Chemistry*, *26*(16), 1668–1688. https://doi.org/10.1002/jcc.20290

155. Martí-Renom, M. A., Stuart, A. C., Fiser, A., Sánchez, R., Melo, F., & Sali, A. (2000). Comparative protein structure modeling of genes and genomes. *Annual Review of Biophysics and Biomolecular Structure*, *29*, 291–325. https://doi.org/10.1146/annurev.biophys.29.1.291

156. Lensink, M. F., & Wodak, S. J. (2013). Docking, scoring, and affinity prediction in CAPRI. *Proteins*, *81*(12), 2082–2095. https://doi.org/10.1002/prot.24428

157. Janin, J. (2010). Protein–protein docking tested in blind predictions: The CAPRI experiment. *Molecular BioSystems*, *6*(12), 2351–2362. https://doi.org/10.1039/C005060C

158. Paxman, J. J., & Heras, B. (2017). Bioinformatics Tools and Resources for Analyzing Protein Structures. Em *Proteome Bioinformatics*. Springer New York.

159. Agirre, J., Atanasova, M., Bagdonas, H., Ballard, C. B., Baslé, A., Beilsten-

Edmands, J., Borges, R. J., Brown, D. G., Burgos-Mármol, J. J., Berrisford, J. M., Bond, P. S., Caballero, I., Catapano, L., Chojnowski, G., Cook, A. G., Cowtan, K. D., Croll, T. I., Debreczeni, J. É., Devenish, N. E., … Yamashita, K. (2023). The CCP4 suite: Integrative software for macromolecular crystallography. *Acta Crystallographica Section D: Structural Biology*, *79*(6), 449–461. https://doi.org/10.1107/S2059798323003595

160. Djaffardjy, M., Marchment, G., Sebe, C., Blanchet, R., Belhajjame, K., Gaignard, A., Lemoine, F., & Cohen-Boulakia, S. (2023). Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. *Computational and Structural Biotechnology Journal*, *21*, 2075–2085. https://doi.org/10.1016/j.csbj.2023.03.003

161. Cohen-Boulakia, S., Belhajjame, K., Collin, O., Chopard, J., Froidevaux, C., Gaignard, A., Hinsen, K., Larmande, P., Bras, Y. L., Lemoine, F., Mareuil, F., Ménager, H., Pradal, C., & Blanchet, C. (2017). Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems*, *75*, 284–298. https://doi.org/10.1016/j.future.2017.01.012

162. Bashari Rad, B., Bhatti, H., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, *173*, 8.

163. Boettiger, C. (2015). An introduction to Docker for reproducible research. *SIGOPS Oper. Syst. Rev.*, *49*, 71–79.

164. Laitos, T. (sem data). *ADVANTAGES OF DOCKER*.

165. Kwon, C., Kim, J., & Ahn, J. (2018). DockerBIO: Web application for efficient use of bioinformatics Docker images. *PeerJ*, *6*, e5954. https://doi.org/10.7717/peerj.5954

166. Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319. https://doi.org/10.1038/nbt.3820

167. Mölder, F., Jablonski, K. P., Letcher, B., Hall, M. B., Tomkins-Tinch, C. H., Sochat, V., Forster, J., Lee, S., Twardziok, S. O., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., & Köster, J. (2021). *Sustainable data analysis with Snakemake* (10:33). F1000Research. https://doi.org/10.12688/f1000research.29032.1

168. Landau, W. M. (2021). The targets R package: A dynamic Make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. *Journal of Open Source Software*, *6*(57), 2959. https://doi.org/10.21105/joss.02959

169. Rocha, S., Vieira, J., Vázquez, N., López-Fernández, H., Fdez-Riverola, F., Reboiro-Jato, M., Sousa, A. D., & Vieira, C. P. (2019). ATXN1 N-terminal region explains the binding differences of wild-type and expanded forms. *BMC Medical Genomics*, *12*, 145.

170. Tubiana, J., Schneidman-Duhovny, D., & Wolfson, H. J. (2022). ScanNet: An interpretable geometric deep learning model for structure-based protein binding site prediction. *Nature Methods*, *19*(6), 730–739. https://doi.org/10.1038/s41592-022-01490-7