

Biological Databases and Database Tools

Final Database Project

Fall 2020

Brian Wiley
Johns Hopkins University
MS Bioinformatics

SubDBplus: A Protein Substrates Database implemented in Neo4j

[Brian J Wiley](#)¹

¹ Masters of Bioinformatics, Johns Hopkins University, Baltimore, MD

ABSTRACT

An important question to ask in drug discovery when targeting a gene or expression of that gene is what effect can my drug have on that protein's downstream and upstream pathways. But how do we start to analyze various downstream pathways if we don't have the entire picture of all the substrates, either proteins, DNA, RNA or small molecules that our protein has or interacts with. More specifically it would be important to know where in our protein, i.e. which domains or residues are specific to each interaction between protein and substrate. I have developed a database that as its primary function will give the researcher, whether they be a molecular biologist, computational chemist, or in lead-to-target drug discovery, a list of substrates for a particular protein of interest. That database, called SubDBplus (SubDB+) is implemented in both a relational MySQL data warehouse as well as the graphing NoSQL database Neo4j so that in addition to substrates, up and downstream pathways can also be queried. The database forms a collaboration between 4 different substrate or binary interactions databases including phosphorylation and dephosphorylation databases, proteolysis, and a non-specific or various interaction types database that includes but not limited to ubiquitinases, transferases, methyltransferases, acetylases, and G-protein coupled receptors. Up until writing this paper, I was not aware of any interfaces that included a simple way to search a protein's substrates and include a graphing model. However during the development of the database and writing the manuscript, I have also discovered a similar database specific to phosphorylation, RegPhos, which is also implemented in MySQL but to my knowledge does not have a graphing layer to the caliber of Neo4j. During the development of SubDB+ and querying of the interface I have learned that ideas for improvements and extended releases of a database come from using the database, seeing where things can be better as well as seeing limitations and where certain functionalities are best supported in different application layers.

1. INTRODUCTION

When it comes to developing novel workflows for determining biological drug targets of disease phenotypes it is imperative to also analyze the networks and cross-talk interactions of those targets. As protein-protein interactions are not an isolated system, it makes for better analysis of off-target drug effects if we have an idea of those proteins which are closely down stream of our targets, the substrates of our protein targets, their substrates, as well as close upstream proteins of our target(s). There are many databases out there for protein-protein interactions, post-translational modifications (PTMs) of protein of interest as well as substrate databases. However, to my knowledge I have not come across

any databases that make it simple to determine all the substrates of a protein, in particular the substrates of kinase and phosphatase families of proteins. Erratic phosphorylation pathways are of primary interest of cancer therapeutics such as kinase inhibitors. According to DrugBank, there are 117 kinase inhibitors in which all are primarily used to treat neoplasms and cancer (DrugBank, 2006). By sourcing a collaboration of substrate and interaction databases, I have designed a multifaceted database infrastructure SubDBplus (SubDB+) that harnesses the power of combining both relational and graphing based databases to deliver a fast simplistic way to determine substrates of kinases and 6 other protein families. By having an clear idea of immediate downstream pathway of kinase and protein inhibitors we can study potential effects for our drugs; therapeutically as well as unintentional side effects.

2. RELATED WORK

As previously stated there are variety of interaction databases for protein-protein, protein-DNA, protein-RNA interactions as well as databases strictly or primarily for storing information on proteins and their substrates. The inspiration for this project comes from a combination of all of their positive attributes. The Reactome Knowledgebase database is an object oriented graph database for pathways and interactions and uses the same graph platform Neo4j and its Cypher querying language as that chosen for the graph implementation of SubDB+. Reactome is built on a much larger scale with around 100,000 proteins, complexes and reactions each as well as around 15,000 pathways (Fabregat et al, Reactome Statistics, 2016). Similarly SubDB+ includes chemical compounds and drugs in the Protein Data Bank (PDB) structures for the selected protein of interest as does Reactome. The difference however despite both linking compounds to CHEBI (SubDB+ also links to DrugBank) is that Reactome does this on the pathway level while SubDB+ does this on the protein level. One critical function that Reactome delivers in which future versions of SubDB+ hope to achieve to be able to characterize and cluster different pathways into associated diseases for the protein-substrate interactions.

Perhaps the most influential protein database for the implementation of SubDB+ was the KinaseNET Human Protein Kinase Knowledgebase. This database is a collaborative of sub-databases developed by Kinexus Bioinformatics Corporation and is an extremely simple designed kinase database that includes structures, substrates, amino acid matrix frequency for determining canonical substrate sequences, upstream kinases, interacting proteins, and regulatory activities and modifications, as well as inhibitors (KinaseNET, Vancouver, British Columbia). The interface is really simple to navigate starting with a search bar for kinase name or UniProt ID number exactly like SubDB+. The point of view for each landing page is from the particular protein kinase perspective, also a common feature of SubDB+. Another critical and almost requirement for accurate and productive scientific research is knowing whether the effect of the protein modification site has a positive or negative influence on the substrate's normal biological function, that is, does the kinase or protein's modification to the substrate allosterically activate or inhibit the substrate's biological function. KinaseNET documents this for a good percentage of their phosphorylation sites both as PTM from upstream kinases on the kinase as

well as that kinase's substrates. Another innovative function of KinaseNET is that for each phosphorylation site both from upstream kinases on the kinase search as well as its substrates is a prediction algorithm Kinase Predictor V2 which will predict with a score the most probable kinases that will contribute to phosphorylation on that specific residue site. KinaseNET also has other daughter/sister databases available from their website such as OncoNET for oncology research and DrugKiNET which focuses on the inhibitors previously mentioned on DrugBank.

All of this really great functionality of KinaseNET does come at a cost. As the company is not publicly funded the information is only available on the KinaseNET website and no API or database download is available. However after reaching to KinaseNET, the president and CSO directed me to another kinase database website created by a former employee named PhosphoSitePlus. This site is similar to KinaseNET with information on downstream substrates and upstream kinases from the kinase of interest perspective. PhosphoSitePlus does supply a bulk download of kinase substrates with their phosphorylation sites from their database and this download of nearly 20,000 kinase-substrate reactions was used to populate the primary of the data for the kinase family of proteins in SubDB+.

Two other large protein interaction/protein substrate databases are both delivered under the umbrella of databases developed by the European Molecular Biology Laboratory – European Bioinformatics Institute (EMBL-EBI). The first one called IntAct Molecular Interaction Database is a repository for binary interactions primarily between two proteins or self-interactions of proteins. This database is also very largely scaled with over 1 million interactions derived from curation and direct experimental submissions (Orchard et al, 2014). UniProtKB protein entries will include a majority of the interactions section listed on their entry page under binary interactions directly linking to the IntAct interaction experiment page. As a closely second major motivation for SubDB+, the UniProt database sparked the interest for a resource that could list the substrates and downstream substrates of a protein or proteins of interest. The UniProt entry for each protein has a section for PTMs in which lists the modifications and modification sites of the protein entry by upstream proteins but when you link to those upstream proteins you again only have the interactions listed and the PTMs of that protein by its upstream modifiers, functionality that is available in the graph portion of SubDB+, however UniProt does not have a fast easy way to list the substrate(s) of protein entry as SubDB+ does. Finally, the MEROPS Peptidase Database is a repository for peptidases or proteases and their substrates as well a division of the platform for inhibitors of peptidases. MEROPS includes peptidases for 32,300 different organisms (some only have 1 peptidase listed) including 884 peptidases and 248 homologs in humans. MEROPS lists the cleavage site of the substrates by the peptidase and the resulting protein fragments from the cleavage. It does this for all known species that have that peptidase however there is no column for the organism name. Substrates are just one of the many tabs of peptidase entry, including by not limited to alignments, inhibitors and pharma (MEROPS, Rawlings et al, 2014).

3. REQUIREMENTS

The requirements for SubDB+ are pretty simple from the user perspective. We want to be able search by gene names or UniProt ID to submit for a list of substrates. In addition to obtaining just the

substrates, it is pretty easy to add in the ability to return a recursive graph search utilizing a graphing database infrastructure. This allows for the user to be able to enter the length of the path and whether the path should recursively traverse downstream, upstream or in both directions. For the substrates information it is imperative to collect data from trustworthy sites and/or databases. In the initial deployment of the database, we have chosen to select 7 different protein families and 8 different species to load proteins from the UniProt database. The protein families are kinases, phosphatases, transferases, methyltransferases, peptidases, isomerases, & G-protein coupled receptors. The species are *Homo sapiens* (human), *Mus musculus* (mouse), *Rattus norvegicus* (rat), *Cricetulus griseus* (Chinese hamster), *Escherichia coli*, *Saccharomyces cerevisiae*, *Arabidopsis thaliana*, & *Drosophila melanogaster* (Fruit fly). In addition to the interaction and substrate requirements, users will be able to review a list of PDB structures for the selected protein of interest as well as the binding sites and drugs that bind these sites in the crystal or NMR PDB structures. The interface for SubDB+ also needed to hyperlink out to the variety of resources from which the data was collected and other possible resources such as the Research Collaboratory for Structural Bioinformatics (RCSB) for PDB structures (Burley et al, 2019), DrugBank and Chemical Entities of Biological Interest (ChEBI) for binding site annotation of these structures, as well as SwissModel for modeling of proteins without known structures submitted.

4. LOGICAL DATABASE DESIGN

The data model for SubDB+ starts off from a protein perspective without any interactions. Information retrieved from the protein databases setup the initial loading of the data warehouse to eventually be loaded into the nodes of the graph. The structure of these proteins is resourced from a Protein Data Bank for each protein in the 7 protein families for all 8 species. Secondly, a list of interaction and substrate databases is compiled which have bulk downloads that must be cleaned as well and performing field manipulation. The information from these databases comprises both new protein level information in addition to the proteins loaded from the first protein loading step if those proteins were not already loaded or updated if already existed as well as the information for the interaction such as detection methods, interaction types, whether the interaction of protein and substrate was detected in vivo, in vitro or both, as well as their source database and literature (See Supplemental file 1 under Appendix A: Data Dictionary for the data warehouse schema). Language drivers are then used to query the data warehouse for loading nodes and interaction edges from proteins to substrates in the graph platform. This makes it simple to query the fields that create the most similar node protein, interaction edge, node substrate data structure as well as the properties which comprise each node and edge. Fig 1. below shows the logical road-map for the SubDB+ database from protein and interaction database input to final interface output. The road-map itself is indeed a “graph” preview of what is in store for the end-user.

5. DATABASE TYPES

I chose to utilize both the relational database management system (RDBMS) as well as the object-oriented database management system (OODBMS) for SubDB+ as the two different types of data that are delivered to end-user could not easily be achieved with just one database type.

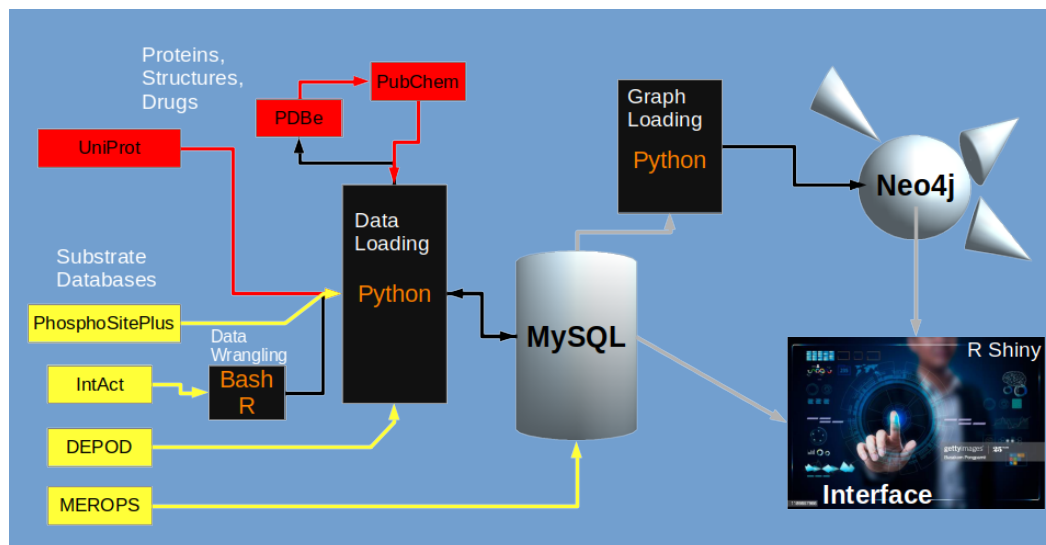


Figure 1. Logical Database road-map for SubDB+. Data loaded into the MySQL data warehouse of SubDB+ can be categorized into two groups, 1) Protein, structures and molecules/drugs that bind those structures (red) and 2) protein-protein interaction and substrate data (yellow). Scripting languages such as R and Python are used to cleanse and modify necessary fields to provide an easier streamline for aligning the most similar possible data structures for graph loading in Neo4j. The R Shiny interface talks to both MySQL warehouse as well as graph and data retrieval from Neo4j.

Despite probably being able to include the protein structure and drug binding site information stored in object documents for each protein such as storing in a JavaScript Object Notation (JSON) document for each protein, this would lead to lengthy documents and expensive querying through non graphing OODBMS systems like MongoDB as some proteins have hundred of structures. Therefore it was chosen to store this information in a MySQL relational database tied to the protein for easy querying by the backend of the interface for the protein of interest. The reason for this is was that there is not much description field information that changes at a high level for listing the molecules that bind proteins and their binding sites. The data grows longer as more structures are determined but as far as the simple functionality of delivering a list protein binding molecules or drugs and their locations, the description stays less or more the same dispersion of required attributes to maintain.

The need for an OODBMS, and particularly a graphing OODBMS, is much more black and white when it comes to determining the extensive and directional dispersion that comes with protein-protein and protein-substrate interactions. For instance, let's say choose a protein of interest, call it "Protein A". Protein A may have 50 substrates where each of those substrates have any number of substrates as

well as other protein interacting on them. If we want the option to view some really unique path, using the simple visual of a tree (Fig. 2), it could quite possibly require a different table for each protein and its substrates and to create a relational join on each protein table as well as the costs to traverse the tables to reach the next protein as a substrate in the table could be substantial. This is why existing protein pathway databases such as KEGG Pathways and Reactome, even for really short pathways, implement a graphing data structure to hold the proteins and their downstream interactors.

The limitations of implementing two database types that underlie the backend to SubDB+ are the storage requirements and cost as well as the run time for loading, updating, curating, and maintaining the data.

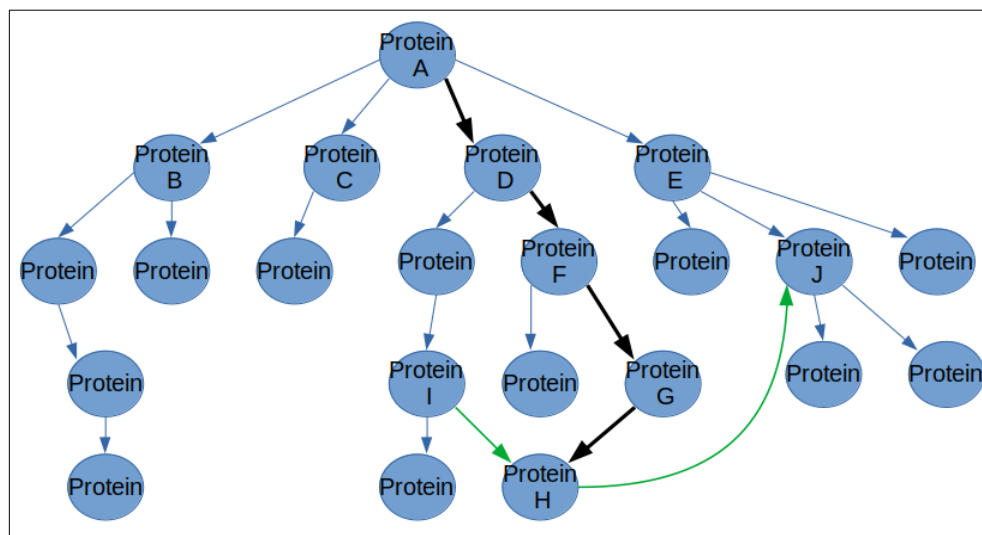


Figure 2. Example of protein-protein interaction tree traversal. Simplified “tree” (as children of tree can only have 1 parent this is a graph) shows a simple substrate path from Protein A to Protein H. In a relational database where each protein is its own table would require searching Protein A’s table for Protein D as substrate, joining Protein D’s table and searching for Protein F, joining Protein F’s table and searching for Protein G, and finally joining Protein G’s table and searching for Protein H. If interested to see H is substrate also of Protein I, we need to join tables from A to I to H and similarly from A to E to J to see common downstream substrate J. With a graph implementation we can traverse the downstream substrates without the requirement for complex join queries.

6. APPLICATION DESIGN

The tables for the MySQL warehouse can be divided into 2 major sections where the proteins section is subdivided into 2 subsections (Fig. 3). The first subsection for proteins is the table that houses the information for the proteins. This information is used to populate nodes in the Neo4j graph. These nodes can potentially be start nodes (blue arrow), i.e. the proteins, end nodes (black arrow), i.e. the substrates or both. The second subsection for the proteins are their structures, their binding sites, and the drugs or molecules that bind them. This information does not enter the graphing database before being viewed from the interface. The second major section, indicated as ‘Substrates Databases’ in yellow in Fig. 3, of the data warehouse stores the interaction and substrate reactions from various downloaded database files supplied by these databases. This information contains 2 proteins in the interaction where 1 protein is the substrate. All substrate databases contain these binary interactors and

so the information from the substrate tables will always populate start and end nodes in the Neo4j for every interaction even if the interaction is a self-interaction which is explained below in the implementation section.

(See supplemental figure in ApplicationDesign.pdf file)

Figure 3. Application Design for SubDB+. Two sections of the data warehouse are colored red for protein information and yellow for substrate information. The blue arrows indicate information used from the resource to populate start nodes (the protein), the black arrow – end nodes (the substrate), and the green arrows depict the source for the interaction between protein and substrate. The structures and binding site details depicted by the red arrow and are queried directly from the MySQL data warehouse for the protein of interest selected in the interface for the end-user to review. The recursive graph query for the protein of interest is performed on the Neo4j database (gray box and gray arrow) and is seen as a visible graph in the interface.

7. IMPLEMENTATION

7.1 DATA WAREHOUSE SOURCING

The initial sourcing of the 7 protein families totaling 8,315 proteins for which will be the auto-completion list of proteins to select in the interface comes from the UniProtKB database of proteins. To source this information we queried the UniProtKB RESTful API for all 8 species, 1 protein family at a time in tabular format with the columns names required to fill the proteins table in the data warehouse. Python was used to access the API. I discovered that from this tabular format the ‘Protein names’ field was hard to parse since some of the protein names had parenthesis in the name and parenthesis were used as a grouping delimiter to separate other alternate protein names which was probably a poor choice by UniProt. It would have been better to use a pipe (|) delimiter as this is not a popular character used in protein names. To distinguish the recommended protein name from the alternate protein names I had to enter the XML format for each protein. As I already had to do this to obtain the information for all PDB structures associated with a protein, this was not really an issue. However, if there is no other reason needed for parsing the XML format this would require extra time that could be avoided with using a different delimiter such as how the Human Proteome Organisation Proteomics Standards Initiative Molecular Interactions (HUPO PSI-MI) developed the PSI-MI TAB Format to use pipe delimiters for multiple field entries (Kerrien et. al, 2007), which was the format used to obtain data download from the first interaction database, IntAct, used for creating SubDB+.

For each PDB structure the Protein Data Bank in Europe REST Entry-base API was accessed, again through Python, for both the ‘ligand_monomers’ and ‘binding_sites’ endpoints for populating a binding sites table as well as the ‘drugbank’ endpoint to populate drug information. I came across two issues

with the PDBe API. First, the 'drugbank' endpoint only had the DrugBank ID and the 2- or 3-digit alphanumeric short name but no longer or preferred name. To solve this issue I went to the PubChem entry (Kim et. al, 2019) for the DrugBank ID in XML format to retrieve the first synonym or preferred drug name. Another issue comes from the fact that a critical point of data, the name for the binding sites that are included usually with "REMARK 800" in the PDB files, were not included in the 'ligand_monomers' endpoint but were included as the sites where the residues bind in the 'binding_sites' endpoint. Therefore there was no error proof way to connect these. I did however find a way to parse the residue number for the binding sites usually included in "REMARK 800" and so this was used instead, however this is not as error proof as using the binding site name, e.g. AC1 or AC2, and some of the details will not include this residue number for various reasons. I have submitted a request to PDBe to have the site names of the PDB file under the 'ligand_monomers' endpoint.

The four interaction/substrate databases used to source the substrate interactions data that would eventually be entered to make the edges in the Neo4j database was collected from IntAct Molecular Interaction Database (Orchard et. al, 2014), MEROPS peptidase Database (Rawlings et. al, 2014), PhosphoSitePlus (Hornbeck et. al, 2014), and the human DEPhOsphorylation Database DEPOD (Damle & Köhn, M., 2019). As previously indicated, IntAct is a binary interaction database and stores bulk interaction data in a clean and organized PSI-MI TAB Format. Downloads of the entire database can made or subsets of curated PSI-MI TAB files for specific disease categories such as Coronavirus, cancer, cardiac related proteins, and neurodegeneration. It is not limited nor is it implemented to easily decipher the directional protein/protein-substrate interaction. Based on advice from IntAct database maintainers there are columns for the 'biological role' for both interactors which is taken from PSI-MI vocabulary. The entire database of interactions, totaling over 1 million binary interactions was first subsetted for interaction where at least one of the interactors was not termed with 'biological role' as 'unspecified role'. This substantially decreased the dataset from over 1 million to just over 10,000 interactions. This subsetted dataset was first cleaned in the R programming language (See additional R file intact_read.R) and based on these columns for 'biological role' a direction was determined. If the biological role for interactor B was blank or interactor ID for both interactors was the same it was a self interaction (direction $A > A$), if either role was an inhibitor then the direction was from the inhibitor to the protein that was being inhibited as well as setting a true negative interaction. There were just 751 of these inhibitory interactions from the subsetted listed of just over 10,000. Obviously this is an issue being able to only determine 751 inhibitory interactions from over 10,000 binary interactions (7.5%) as prior biological research indicates there are many regulatory inhibitory protein-protein interactions. Lastly, if biological roles of either interactor included 'target' or 'acceptor' the direction was determined to be from the other protein to the target or acceptor.

For loading the data for the second substrates database, the MEROPS peptidase database, this database include a download of all the SQL files needed to recreate all or mostly all the information include in MEROPS. This includes over 70 scripts however the most critical script for SubDB+ was for a 'Substrate_search' SQL script which was taken and loaded into the data warehouse without any changes. This included nearly 100,000 proteolytic reactions for all species in addition to those chosen for SubDB+. There were 2 minor issues I had come across during the SubDB+ implementation with this table from MEROPS. First was that despite there being a column for the UniProt ID for the substrate, there was no Uniprot ID column for the protease. As a solution to this problem, since there

was a MEROP ID ‘code’ column, I used the MEROPS ID cross-reference column from UniProt for populating this information. More about this solution is discussed below in Neo4j Graph Loading. A second issue was there was no preferred gene name column or that there like and so with all the UniProt IDs for the substrates using a batch request payload (UniProt_programmatically_py3, 2017), I created an auxillary table for just the MEROPS substrates.

The third and fourth substrate database, PhosphoSitePlus (PSP) and human DPhosphorylation Database (DEPOD), are molecularly related in that they both source information on the phosphorylation of dephosphorylation of proteins respectively. PSP, a cousin database to KinaseNET, also details other types of PTMs including but not limited to acetylation, methylation, and ubiquitination sites of proteins however there is not any information on which proteins create these other PTMs, i.e. which proteins are the entries substrates of, as there is for the phosphorylation reactions. There is a one bulk dataset for substrate interactions that was downloaded from PSP, called Kinase_Substrate_Dataset, that was used to populate its own table in the data warehouse containing just over 19,000 site-specific kinase/kinase-substrate reactions. Similarly for DEPOD, a database that exclusively reports information for human phosphatases, includes a download of nearly 1,000 phosphatase/phosphatase-substrate protein pairs. Some of the pairs have multiple sites of dephosphorylation however some of the pairs do not have a site, i.e. the site of the dephosphorylation on the substrate is not known. There is no way to handle the site not being known but it is still important information to know which substrates a phosphatase has and so this information was still added to the data warehouse.

7.2 NEO4J GRAPH LOADING

Data loading from the MySQL data warehouse into the Neo4j graph serves as the key important implementation for the substrates and substrate pathway requirements of SubDB+. It combines details from the protein perspective for all protein and substrate tables in the data warehouse without the need for any complex join statements. First task was entering the 8,315 proteins as nodes into the graph for the 7 protein families. At this point the nodes are not determined to be start nodes or end nodes of a relationship. The Python MySQL driver was used to query the necessary attributes for visualizing in Neo4j and the interface, including UniProt ID, protein name, organism and taxonomy ID, the primary gene name as the visible label for the node, as well as alternate gene and protein names. This information was then entered into Neo4j again with the Python driver for Neo4j.

Entering the substrate interaction data as the edges from protein to substrate is slightly more complicated than the initial protein loading. All of the substrate database both had protein and substrates included and not included in the initial 8,315 proteins for the 7 families sourced from UniProt. Therefore, while entering the interaction data as the edges from start node to end node, if the protein existed already as a node it was not recreated but updated or appended with additional naming information and if it was not already a node in the graph, then a new protein, or molecule as some IntAct data includes molecules in addition to proteins, node was created. The most difficult challenge came with choosing what attributes of a protein/protein-substrate interaction were going to be used and the names for these attributes in the edges of the graph that would lead to the most consistent edges

data structure for the data queried of all 4 substrate databases. For instance for proteolytic reactions there is usually two protein molecules for the substrate that result from the protease cleavage and so as this is stored as a substrate formula in the MEROPS, this was used to populate the site(s) attribute in the edge between protein node and substrate node. However the sites of phosphorylation and dephosphorylation for PSP and DEPOD respectively were used to populate this field. After further analyzing the MEROPS table, I had discovered there is also a substrate residue number field that is the residue number for the substrate in UniProt, and this field seems more appropriate for the site(s) attribute and an additional field for the formula should be entered for MEROPS reactions. Another example is that the bulk substrate data download from the PSP site did not have a field for literature despite the entries on their interface having them. As this field was available from IntAct and DEPOD downloads and the MEROPS SQL script, it was only natural to include a literature field in the edge between protein and substrate nodes and to leave this blank for PSP.

7.3 INTERFACE SEARCH

The interface for SubDB+ was implemented as a webpage in R Shiny and uses both MySQL driver and the “neo4r” R driver for Neo4j to source the data (red and gray arrows in Fig. 3 respectively). The interface can be broken into two major sections, a search section and an SQL section. The search section (Fig. 4) leads to subsections that included the Graph for the protein searched, a DataTables (Jardine, 2014) representation of the individual interactions and substrate sites in the Graph, a Substrate table listing every single substrate in SubDB+ for the searched protein, as well as tabs for PDB Structures and PDB Binding Sites and Drugs for the protein searched which is sourced from the data warehouse. The search page allows for selecting for a recursive graph search, the direction of the graph in relation to the protein searched, i.e. down for downstream reactions and up for upstream reactions, and the length of the interaction pathway. The degenerate default case is set to direction of down and pathway length of 1 for the protein’s substrates. A limit is also able to be selected to limit the amount of reactions returned in the graph. As the UI graph is implemented with Plotly and the nodes cannot be maneuvered around like in the Neo4j graphing database, I decided to limit this to 200 reactions in the interface. Because of this, if the user selects, for instance, both directions for upstream and downstream pathways with a longer pathway length, e.g. length of 4, then the extent of the recursive search is limited to further (deeper) instances on the recursion stack and not many direct substrates are listed, however this may bring up interesting pathway targets and relations.

Search

Filters

Protein Family:
kinase, peptidase, phosphatase, G-protein ▼

Organism:
Homo sapiens, Drosophila melanogaster, ▼

Gene Name: (type to choose IDs)
Examples: CDK7, PRKACA, Casp3, BACE1

UniProt ID:
Examples: Q00311, P17612, P70677, P56817, Q8BYF1

For substrates direction=down & length=1

Path Direction
down ▼

Path Length
1 ▼

Limit Relationship Results to:
10

Show 10 entries

Search:

Get Graph

Head Protein Family	Organism	Gene Name	UniProt ID
No data available in table			

Showing 0 to 0 of 0 entries

Previous Next

Figure 4. Search landing page of SubDB+. The auto-complete search fields are in the red box. User can filter what auto-completes to the left and select the direction and length of pathway from the selected protein of interest.

As a use case we will search the popular molecular kinase target for drug discovery, Protein Kinase A, gene name PRKACA (PKA). PKA is kinase involved in many, many pathways and has many substrates involved in glucose transport and insulin regulation, calcium transport, nuclear targets as well as proteins involved the cell cycle, mitogens and therefore cancer. We can search either by the gene name or by the UniProt ID (Fig. 5A). After clicking for the graph the user can hover and maneuver to and zoom in on certain regions (Fig. 5B).

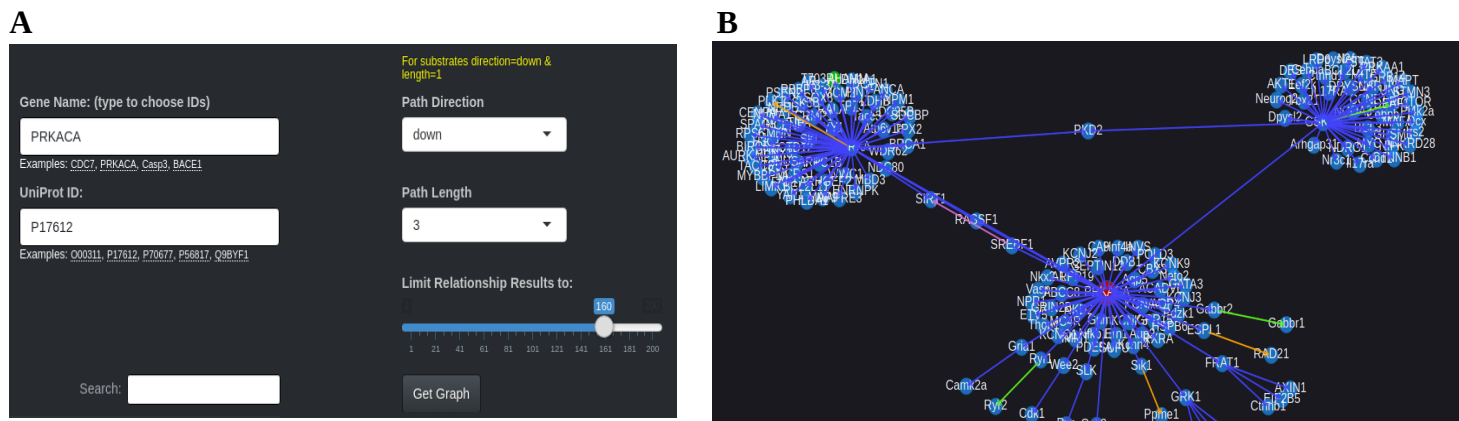


Figure 5. Search and Graph interface. **A)** The search landing page in SubDB+. Not shown but to the left are filters for species and head protein family which filters the auto-completion fields for Gene Name and UniProt ID. **B)** An example graph showing for PKA by selecting only down stream pathways from PKA with length between 1 and 3 proteins in path.

The selected protein PKA will be in red (Fig. 6A). The user can hover over the nodes of the proteins and their substrates as well as the edges between them. Plotly is not the best package for being able to hover along the entire edge to see the information and so the user must hover over the arrow head at the substrate to see this (Fig. 6A) as well as not being able to move nodes and edges out of the way for better visibility. Alternatively, there will be a graph table which summarizes every individual reaction in the graph as well as the most important “Substrates” tab that is specifically a downloadable list of all the reactions in the graph 1 interaction downstream of the selected protein (Fig. 6B & 6C). Both graph table and substrate table link to the substrate’s UniProt entry as well as PSP for both kinase and substrate if the reaction was sourced from PSP (Fig. 6D).

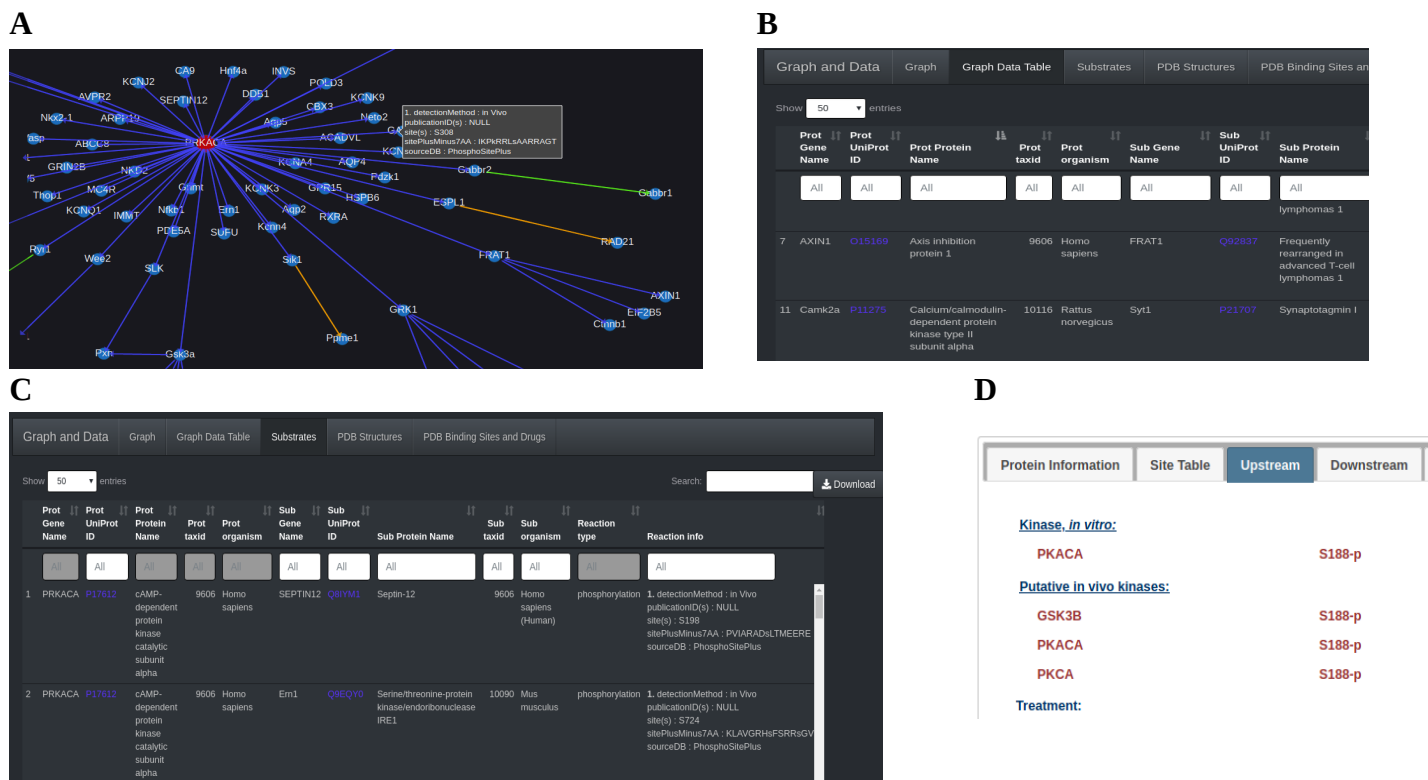


Figure 6. The results from protein search of SubDB+. **A**) Plotly does allow for zooming in on specific portion of the graph and hovering over nodes for node properties and arrow heads for protein substrate reaction info. **B**) Table representation for all entries in the graph **C**) List of all substrates for the selected protein. This list is also filterable and downloadable **D**) PSP page linked from UniProt ID for a substrate in which the reaction is sourced from PSP.

Additional tabs to view from a protein search include the PDB binding sites. One of the drugs that binds and inhibits PKA is H-89. SubDB+ will link to the entry for H-89 at DrugBank where other targets can be seen (Fig. 7B). Another target of H-89 is the kinase HASPIN. Because we had limited the graph search for display purposes, we can redo a search with HASPIN (Fig. 7C) or submit a Cypher query in the Neo4j database to include a start protein PKA and an end protein HASPIN or more than 2 proteins and search for those paths based on restrictions. This will be implemented in a future release of SubDB+ (See next section in Future Work). We can see from the Cypher query that PKA and HASPIN are related through CDK1 (Fig. 7D) as well as protein kinase aurora-B (AURKB) and a phosphatase PPP2CB (Fig. 7D). Therefore, we could see some off target effects in treating PKA with H-89.

A

Graph and Data

Graph

Graph Data Table

Substrates

PDB Structures

PDB Binding Sites and Drugs

Show50entries

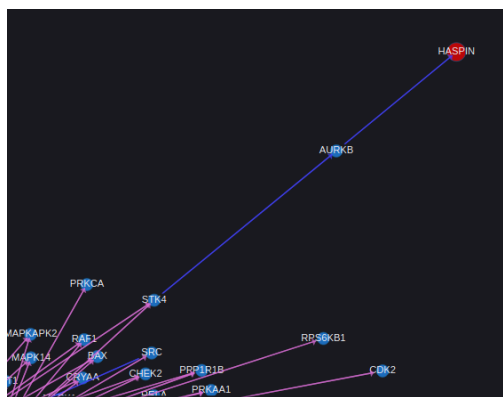
Search:

UniProt Protein Chain	PDB ID	PDB Site ID	Structure Residue #	UniProt Residue #	Residue	Residue Chain	Ligand Residue #	Ligand Short	Ligand Long	Ligand Chain	DrugBank ID
All	All	All	All	All	All	All	All	IQB	All	All	All
477 A	3VOH	AC1	71	70	ALA	A	401	IQB	N-[2-(4-BROMOCINNAMYLAMINO)ETHYL]-5-ISOQUINOLINE SULFONAMIDE	A	DB07995
475 A	3VOH	AC1	57	56	ARG	A	401	IQB	N-[2-(4-BROMOCINNAMYLAMINO)ETHYL]-5-ISOQUINOLINE SULFONAMIDE	A	DB07995

B

Pharmacodynamics	Not Available
Mechanism of action	<p>TARGET</p> <ul style="list-style-type: none"> cAMP-dependent protein kinase catalytic subunit alpha cAMP-dependent protein kinase inhibitor alpha Serine/threonine-protein kinase haspin

C



D

```
//Multiple Protein Search
MATCH p = (n:Protein {name:'PRKACA'})<--[:INTERACTION*1..3]-->(b:Protein {name:'HASPIN'})<--[:INTERACTION*1..3]-->(c:Protein {name:'PPP2CB'})
WITH *, relationships(p) as rs
UNWIND rs as relationship
RETURN
    startNode(relationship).name as Protein1,
    relationship['name'] AS 'Interaction type',
    endNode(relationship).name as Protein2, length(p), p
LIMIT 50
```

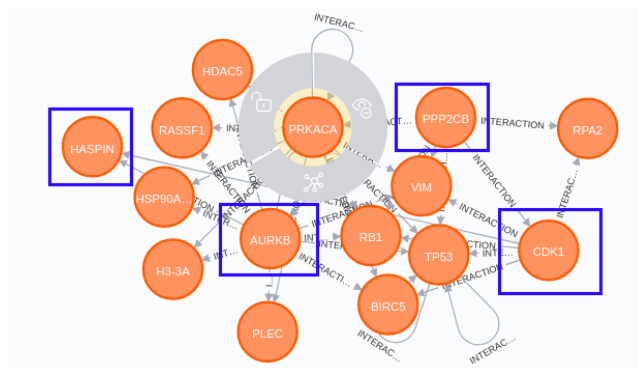



Figure 7. Applications from searching protein in SubDB+. **A)** Tab for binding sites and the drugs or molecules that bind them. **B)** Link in to DrugBank from the binding sites tab **C)** Additional follow up search for the other targets of drug used to inhibit PKA **D)** Cypher query using multiple proteins and resulting graph.

Additionally from the PDB binding sites tab and PDB structures tabs we can see the structure inside SubDB+ as well as link directly to the RCSB Protein Data Bank. The binding sites tabs indicated an alanine at residue 70 among other residues that bind H-89 and so we can quickly view the region of the structure for this binding site thanks to the PDB binding sites tab (Fig. 7A and 8B).

A

Graph and Data			
Graph	Graph Data Table	Substrates	PDB Structures
<div> <div>Show 25 entries</div> <div>Search:</div> </div> <div> <div>3D Structure</div>  </div>			
uniProtID	pdbID	chain	
All	All	All	
10 P17612	3OX3	A	
11 P17612	3OOG	A	
12 P17612	3OVV	A	
13 P17612	3OWP	A	
14 P17612	3OX7	A	
15 P17612	3POM	A	
16 P17612	3POO	A	
17 P17612	4AEB	A	
18 P17612	4AEB	A/B	
19 P17612	4AEB	A/B	

B

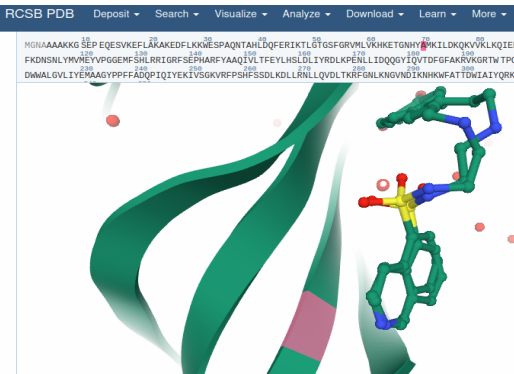


Figure 8. PDB Structures and links to RCSB. **A)** The PDB Structures tab allows for quick glance at a structure. **B)** Link to the structure at RCSB for better visualizing the residues listed in the PDB Binding Sites and Drugs tab.

Finally two other functionalities available in the interface for SubDB+ include a front-end SQL query to search and download data from the data warehouse as well as a Swiss-Model link for predicting structures for protein with no known experimental structures. A place holder shows the query to obtain the kinase with the most unique substrates. In an example we can search for kinases from the PSP table where the kinase/kinase-substrate reactions that also do not have a structure stored in SubDB+ (Fig. 9A), we see one protein involved in cell cycle regulation Wee2 (mouse), which is also a substrate of PKA in mouse has no structures in SubDB+ and so we can model this structure by clicking the link from the PDB structures tab to Swiss-Model (Fig. 9D & 9E).

A

Search MySQL Database

SQL

Table Schema & SQL

Results

Show 50 entries

Search:

Element Name

Table Name

Description

Source

Data Type

Node or Edge

Neo4j Elem. Name

All

[*]uniprotPdbJ

All

All

All

All

41

inVivo

kinasePhosphoSitePlus

Whether databas...

PhosphoSitePlus

tinyint(1)

Edge

detectionMethod

42

inVivo

kinasePhosphoSitePlus

Whether databas...

PhosphoSitePlus

tinyint(1)

Edge

detectionMethod

SQL Query

SELECT DISTINCT uniProtIDKin, geneNamePreferredKin FROM kinasePhosphoSitePlus WHERE kinasePhosphoSitePlus.uniProtIDKin NOT IN (SELECT DISTINCT uniProtID FROM uniprotPdbJoin)

```
SELECT DISTINCT uniProtIDKin, geneNamePreferredKin
FROM kinasePhosphoSitePlus
WHERE kinasePhosphoSitePlus.uniProtIDKin NOT IN
(SELECT DISTINCT uniProtID FROM uniprotPdbJoin)
```

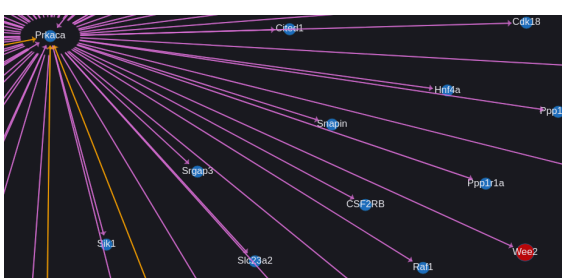
B

SQLTable Schema & SQLResults

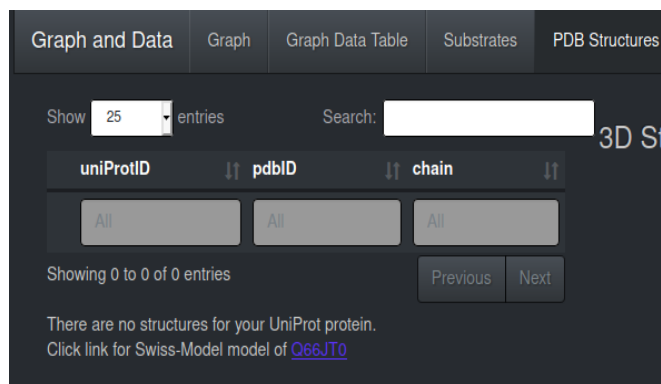
Show50entries

	uniProtIDKin	geneNamePreferredKin	source
	All	All	All
13	Q66JT0	Wee2	PSP
268	P47810	Wee1	PSP
6	O80X41	Vrk1	PSP

C



D



E

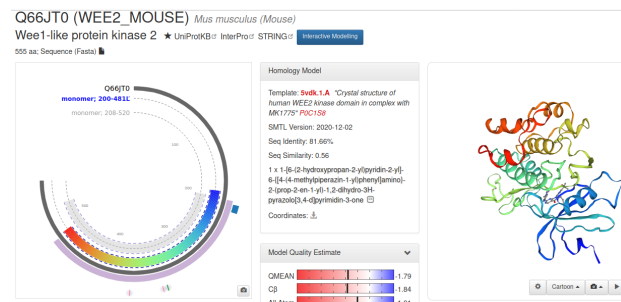


Figure 9. SQL search and structural modeling. **A)** Schema for the data warehouse is listed to allow for users to query without needing a copy of the database or MySQL installed. **B)** Results from a query. **C)** Example search of a selected protein Wee2 from the query. **D)** Since no structures may be available from the database as it was an additional protein than the initial 8,315 there will be a link to **E)** a Swiss-Model structural alignment prediction.

8. CONCLUSION AND FUTURE WORK

The goal of SubDB+ was to simplify and integrate multiple reaction databases so the user can look up any protein to obtain a list of substrates. The motivation for this database started with just kinase substrates from the framework at KinaseNET. However it can easily be extended to all protein families with the right resources. The initial reaction types deployed in SubDB+ are phosphorylation, dephosphorylation, proteolysis, as well as other reaction types from IntAct in which the top reactions not just mentioned are ubiquitination and methylation reactions. The user can simply enter the gene name or UniProt ID for the protein of interest and can obtain a list of substrates. In addition to substrates, graph pathways are also returned and can be customized to include directional pathways and pathway lengths. A list of protein structures including their binding sites and the drugs and molecules that bind them can also be browsed for the selected proteins with links to their respective resources.

For future work I propose performing updates to SubDB+ in 3 phases. The first phase includes make the database public on a server for both the interface as well as the Neo4j browser so users can create their own Cypher queries as well as adding more data to the database. This would include adding more species and more protein families, however more research needs to be performed on data resources for interactions for these other protein families or data mining medical websites and journals. Currently the database only has ~18,500 proteins, ~1,000 molecules and ~31,000 binary interacting pairs with 1 or more reaction sites. As the database grows, I would also like the ability for users to submit their own interactions with experimental evidence that would initially be set as “not reviewed”. The second phase of the database would be to recruit volunteers and students to help curate all entries that are being manually submitted. This would be an ongoing phase that is constantly being updated. The third phase would also contain 2 parts. Part 1 of the third phase would be to transfer the multiple protein search capabilities of Cypher querying in Neo4j similar to the protein network search of RegPhos (Fig. 10) so users can search more specific pathways including identifying a start node of the pathway and end node. In order to achieve this the highly functioning graph visualization the interface would first need

to be overhauled to provide a better infrastructure as indicated Plotly is not the best for graphs. Part 2 would also be continuous in beautifying the interface or adding better linking functionality. This may or may not include rewriting the interface with one or more specific JavaScript frameworks like Node.js, Vue.js, and/or React.js. It would include linking the substrates from a protein search using an little graph icon to search specific substrate graphs, updating the data warehouse with the shortName XML attribute from UniProt, and adding search functionality to accommodate searching short, alternate, and full protein names, e.g. instead of just BACE1, being able to additionally search Beta-secretase 1, Aspartyl protease 2, and Asp 2.

As another future project and sister database that I would like to implement is that in addition to having the list of PDB structures for the searched protein of interest, having the ability to select a structure of protein and any structure from any and all of the substrates and perform computational docking so the user can see the docking sites in 3-dimensions. This is similar idea to how Fold & Function Assignment (FFAS) after predicting folds of a protein sequence links to their ProtMod database from the template and performs modeling with either SCWRL or Modeller. Instead for SubDB+ after selecting PDB structure for protein and substrate will link to the SubDBDock database to run docking scripts at minimum with Chimera and Dock6 from University of California San Francisco.

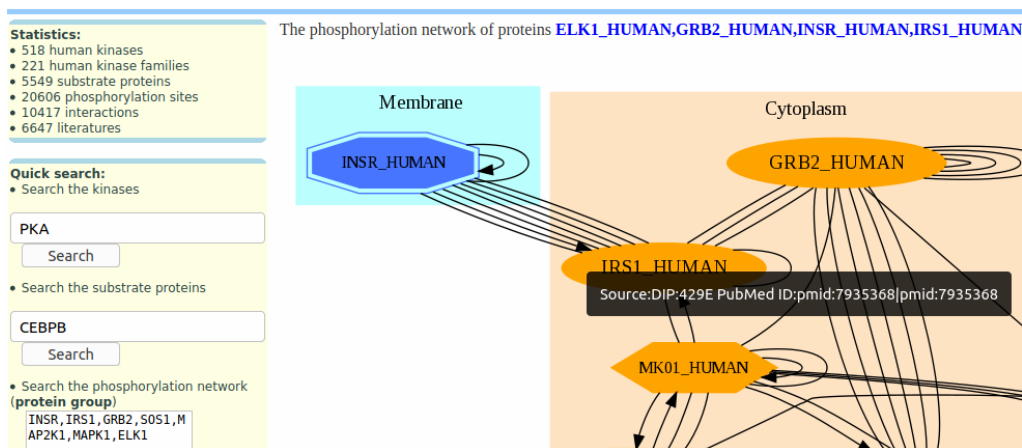


Figure 10. Interface for RegPhos. In the bottoms left of the figure the users are able to enter a series of proteins in a phosphorylation network and can visualize this network in a graph.

9. SUPPLEMENTAL MATERIAL

Appendix A: Data Dictionary

data_dict.csv in Project main folder

Link to Database Demonstration:

<https://www.youtube.com/watch?v=8ew1xJZf7tY>

REFERENCES

1. The UniProt Consortium, UniProt: a worldwide hub of protein knowledge, *Nucleic Acids Research*, Volume 47, Issue D1, 08 January 2019, Pages D506–D515, <https://doi.org/10.1093/nar/gky1049>
2. PDBe-KB consortium, PDBe-KB: a community-driven resource for structural and functional annotations, *Nucleic Acids Research*, Volume 48, Issue D1, 08 January 2020, Pages D344–D353, <https://doi.org/10.1093/nar/gkz853>
3. Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., & Bolton, E. E. (2019). PubChem 2019 update: improved access to chemical data. *Nucleic acids research*, 47(D1), D1102–D1109. <https://doi.org/10.1093/nar/gky1033>
4. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Res.* 2006 Jan 1;34 (Database issue):D668-72. 16381955. Retrieved from <https://go.drugbank.com/categories/DBCAT000737>
5. Rawlings, N. D., Waller, M., Barrett, A. J., & Bateman, A. (2014). MEROPS: the database of proteolytic enzymes, their substrates and inhibitors. *Nucleic acids research*, 42(Database issue), D503–D509. <https://doi.org/10.1093/nar/gkt953>
6. Hornbeck PV, Zhang B, Murray B, Kornhauser JM, Latham V, Skrzypek E PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. *Nucleic Acids Res.* 2015 43:D512-20.
7. Orchard, S., Ammari, M., Aranda, B., Breuza, L., Briganti, L., Broackes-Carter, F., Campbell, N. H., Chavali, G., Chen, C., del-Toro, N., Duesbury, M., Dumousseau, M., Galeota, E., Hinz, U., Iannuccelli, M., Jagannathan, S., Jimenez, R., Khadake, J., Lagreid, A., Licata, L., ... Hermjakob, H. (2014). The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases. *Nucleic acids research*, 42(Database issue), D358–D363. <https://doi.org/10.1093/nar/gkt1115>
8. Damle, N. P., & Köhn, M. (2019). The human DPhOosphorylation Database DEPOD: 2019 update. *Database : the journal of biological databases and curation*, 2019, baz133. <https://doi.org/10.1093/database/baz133>
9. Andrew Waterhouse, Martino Bertoni, Stefan Bienert, Gabriel Studer, Gerardo Tauriello, Rafal Gumieny, Florian T Heer, Tjaart A P de Beer, Christine Rempfer, Lorenza Bordoli, Rosalba Lepore, Torsten Schwede, SWISS-MODEL: homology modelling of protein structures and complexes, *Nucleic Acids Research*, Volume 46, Issue W1, 2 July 2018, Pages W296–W303, <https://doi.org/10.1093/nar/gky427>
10. Fabregat, A., Sidiropoulos, K., Garapati, P., Gillespie, M., Hausmann, K., Haw, R., Jassal, B., Jupe, S., Korninger, F., McKay, S., Matthews, L., May, B., Milacic, M., Rothfels, K., Shamovsky, V., Webber, M., Weiser, J., Williams, M., Wu, G., Stein, L., ... D'Eustachio, P. (2016). The Reactome pathway Knowledgebase. *Nucleic acids research*, 44(D1), D481–D487. <https://doi.org/10.1093/nar/gkv1351>

11. Reactome Statistics. The Reactome pathway Knowledgebase. Retrieved from <https://reactome.org/about/statistics>
12. KinaseNET. Kinexus Bioinformatics Corporation. <http://www.kinaset.net.ca/>
13. Kerrien, S., Orchard, S., Montecchi-Palazzi, L., Aranda, B., Quinn, A. F., Vinod, N., Bader, G. D., Xenarios, I., Wojcik, J., Sherman, D., Tyers, M., Salama, J. J., Moore, S., Ceol, A., Chatr-Aryamontri, A., Oesterheld, M., Stümpflen, V., Salwinski, L., Nerothin, J., Cerami, E., ... Hermjakob, H. (2007). Broadening the horizon--level 2.5 of the HUPO-PSI format for molecular interactions. *BMC biology*, 5, 44. <https://doi.org/10.1186/1741-7007-5-44>
14. EMBL-EBI Training Online. UniProt_programmatically_py3. 2017. Retrieved from https://www.ebi.ac.uk/training/online/sites/ebi.ac.uk.training.online/files/UniProt_programmatically_py3.pdf
15. Stephen K Burley, Helen M. Berman, et al. RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy (2019) *Nucleic Acids Research* 47: D464–D474. doi: [10.1093/nar/gky1004](https://doi.org/10.1093/nar/gky1004). [rcsb.org](https://www.rcsb.org)

Software Used & References where applicable (Python, R, DataTables, CSS, JavaScript, MySQL, neo4j)

1. Neo4j Desktop – Version 1.3.8 and Neo4j Browser
2. Python Version 3.7.6
3. MySQL: Ver 8.0.22-0ubuntu0.20.04.2 for Linux on x86_64
4. Nicholas Rego, David Koes, 3Dmol.js: molecular visualization with WebGL, *Bioinformatics*, Volume 31, Issue 8, 15 April 2015, Pages 1322–1324, <https://doi.org/10.1093/bioinformatics/btu829>
5. R version 4.0.3 (2020-10-10)
6. Jardine, A. (2008-2014). DataTables. Retrieved 01 April, 2014, from <http://datatables.net>
7. Richard Kunze and Mirjam Rehr (2020). dqshiny: Enhance Shiny Apps with Customizable Modules. R package version 0.0.5. <https://github.com/daqana/dqshiny>
8. Jeroen Ooms, David James, Saikat DebRoy, Hadley Wickham and Jeffrey Horner (2020). RMySQL: Database Interface and 'MySQL' Driver for R. R package version 0.10.20. <https://CRAN.R-project.org/package=RMySQL>
9. R Special Interest Group on Databases (R-SIG-DB), Hadley Wickham and Kirill Müller (2019). DBI: R Database Interface. R package version 1.1.0. <https://CRAN.R-project.org/package=DBI>
10. Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2020). shiny: Web Application Framework for R. R package version 1.5.0. <https://CRAN.R-project.org/package=shiny>

11. Winston Chang (2018). shinythemes: Themes for Shiny. R package version 1.1.2. <https://CRAN.R-project.org/package=shinythemes>
12. Victor Perrier, Fanny Meyer and David Granjon (2020). shinyWidgets: Custom Inputs Widgets for Shiny. R package version 0.5.4. <https://CRAN.R-project.org/package=shinyWidgets>
13. Yihui Xie, Joe Cheng and Xianying Tan (2020). DT: A Wrapper of the JavaScript Library 'DataTables'. R package version 0.16. <https://CRAN.R-project.org/package=DT>
14. Dean Attali (2020). shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. R package version 2.0.0. <https://CRAN.R-project.org/package=shinyjs>
15. C. Sievert. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/ CRC Florida, 2020.
16. Gabor Csardi (2015). igraphdata: A Collection of Network Data Sets for the 'igraph' Package. R package version 1.0.1. <https://CRAN.R-project.org/package=igraphdata>
17. Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2020). dplyr: A Grammar of Data Manipulation. R package version 1.0.2. <https://CRAN.R-project.org/package=dplyr>
18. Jeroen Ooms (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. arXiv:1403.2805 [stat.CO] URL <https://arxiv.org/abs/1403.2805>.
19. Rinker, T. W. (2017). qdapRegex: Regular Expression Removal, Extraction, and Replacement Tools. 0.7.2. University at Buffalo. Buffalo, New York. <http://github.com/trinker/qdapRegex>
20. Eric Bailey (2015). shinyBS: Twitter Bootstrap Components for Shiny. R package version 0.61. <https://CRAN.R-project.org/package=shinyBS>
21. Colin Fay (2020). neo4r: A 'Neo4J' Driver. R package version 4.0.0. <https://github.com/neo4j-rstats/neo4r>
22. Hadley Wickham (2020). httr: Tools for Working with URLs and HTTP. R package version 1.4.2. <https://CRAN.R-project.org/package=httr>
23. Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. <https://igraph.org>
24. Kun Ren (2016). rlist: A Toolbox for Non-Tabular Data Manipulation. R package version 0.4.6.1. <https://CRAN.R-project.org/package=rlist>