2020

PhosphoView

SOFTWARE DEVELOPMENT GROUP PROJECT (MSC BIOINFORMATICS)

MOHAMED ABDIKADAR GOLAID, ANASTASIA PAVLOVETS, ZI HAN OOI, SHERIDAN STANLEY, ALISHA ANGDEMBE

Table of Contents

About

PhosphoView is a web application that provides a holistic overview of human protein kinases. We provide background information, namely regarding the kinase-substrate-phosphosite relationship. We also provide users kinase Inhibitor information in the form of drug names and structural properties. We return each feature in a user-friendly way, namely by giving them the option to download our data. One of the principal components of our website is the ability to allow users to upload their own experimental datasets and specific threshold parameters. From this, we return the output in four key components. These include two volcano plots, which display protein Summary and kinase Activity. Also, we provide Kinase Substrate Enrichment plots and score data.

We utilised a number of websites, namely, Ensembl, UniProt, PhosphoSitePlus and PKIDB and incorporated their information into our database using web scraping and APIs. The data was populated in our database using SQLALchemy with SQlite. Our application is built using FLASK, a micro web framework written using Python 3.6. The website is then deployed through Amazon Web Services' Elastic Beanstalk.

Software Overview

Software architecture

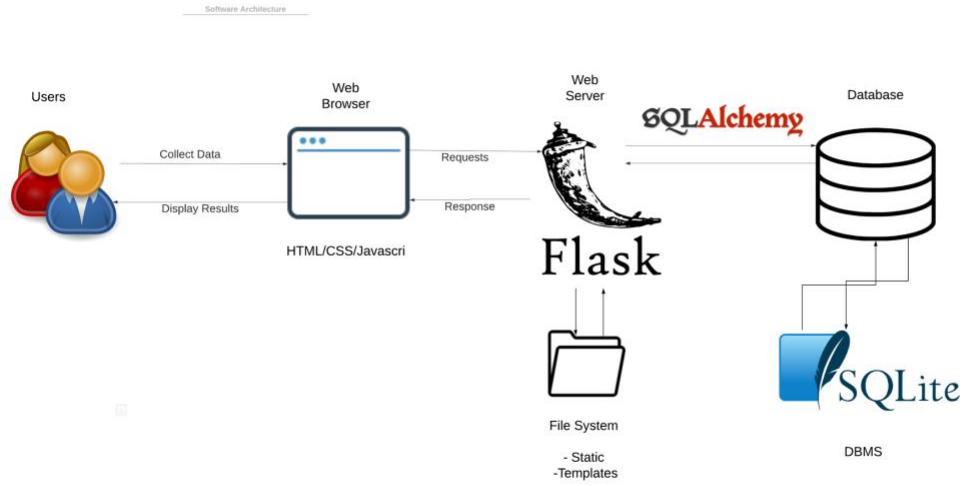

Figure 1. MVC1 PhosphoView software architecture. This schematic illustration highlights the different types software used from Client to server to database.

As highlighted in Figure 1, our web application was grounded in a SQLite3 database. We populated this database with information on kinases, phosphosites and inhibitors. We used SQLAlchemy in order to connect the database with Flask. Flask is a Python microframework which is able to communicate to our file system which has our templates and static folder. This provided a seamless integration between HTML, CSS and JavaScript. Collectively this was then used to render the final website for users.

Website architecture

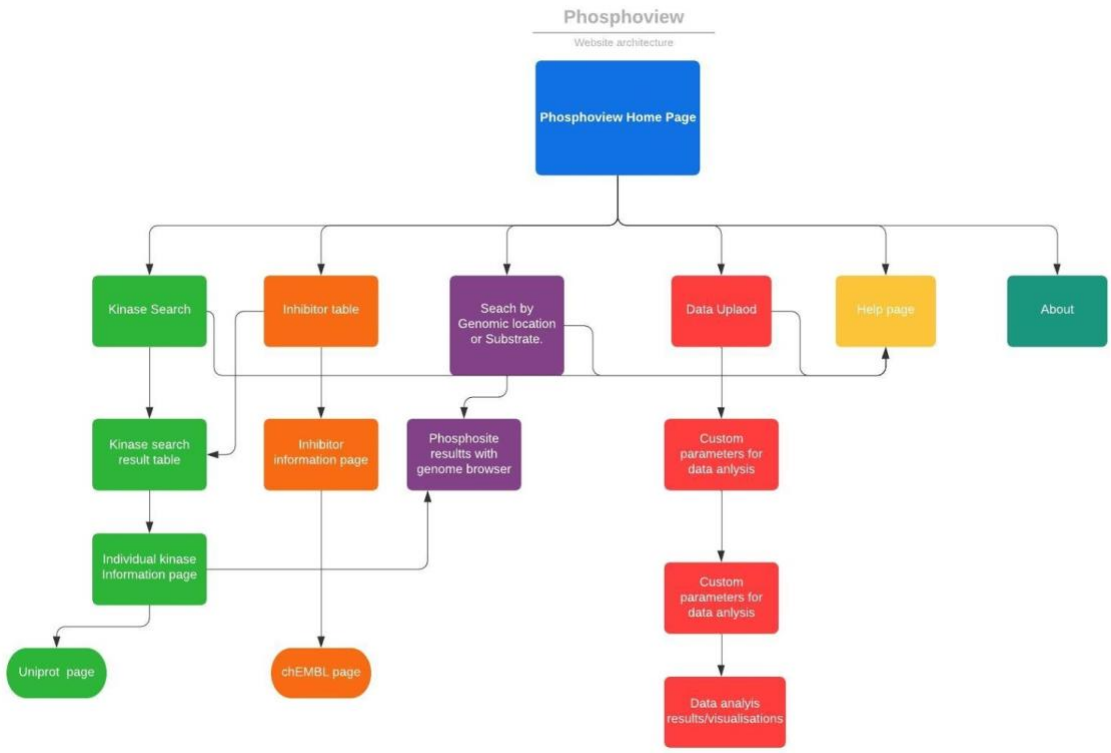

Figure 2. Website architecture of PhosphoView . The squares represent an individual page within the website while the ovals represent a page in an external website. The arrows represent the links that are possible between all pages.

Design philosophy

The design of the web application was inspired by Model-View-Controller (MVC) paradigm commonly used in software design. VMC is a way to internally separate the codes into parts so that they are more organized and easier to share. It also allows task division to be more efficient. The Model is normally the component that contains the database or the closest component adjacent to the database. In the design of this project, Database is separated from the Model. The Model contains codes to access the database, allowing organization and manipulation of the data.

The View is the part that returns the information back to the user, in some form of graphical visualisations such as table, diagrams or charts. Lastly, the Controller is the liaison between the Model and the View. The Controller retrieves the inputs from the user and turns them into requests for the Model. Using MVC, the user never has direct access to the database. This prevents the database from being corrupted.

In our version of web application, we have two MVC flows because one part of the software application deals with users' uploaded document in order to return graphical representations.

Advantages of MVC

- easier multiple programmers to coordinate and work together.
- reduces the load for one person.
- faster development as multiple stages could be developed simultaneously.

Disadvantage of MVC

- needs clear boundaries of the task's division in the beginning
- MVC architecture is not the easiest to understand for junior programmers.

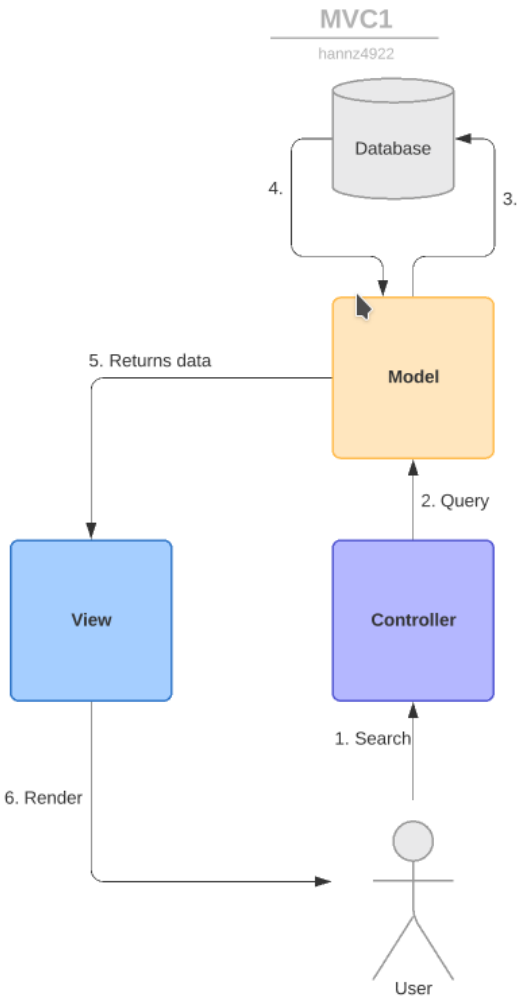

Figure 3. MVC1. The figure shows the first MVC flow in the design of our website.

In the first MVC, the user puts would search for something which is then passed to the Controller. The controller uses functions from the Model to query the database before returning the data back to the View. The View would then render the information into user-friendly format.

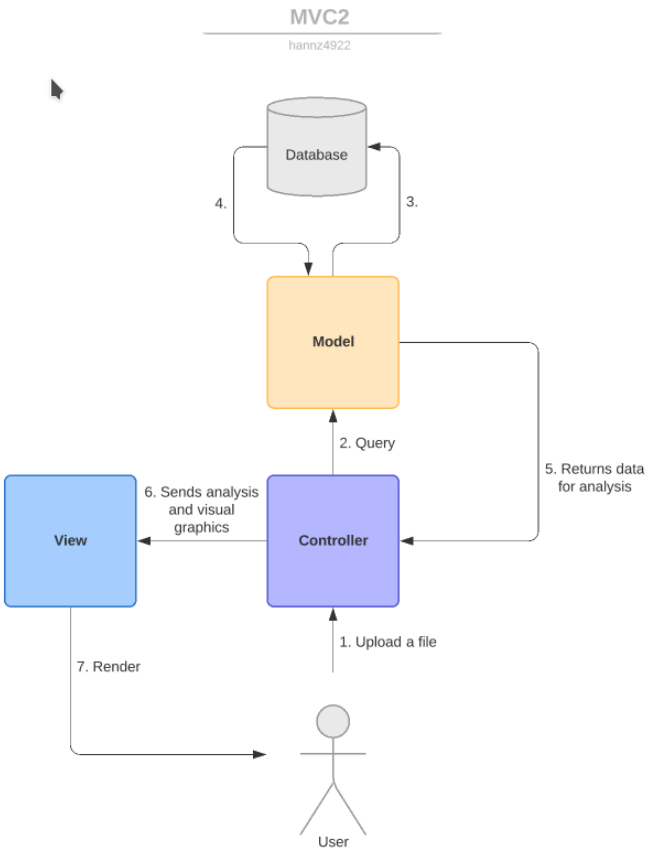

Figure 4. MVC2. The Figure shows the second MVC flow for our website.

In the second part of the MVC, the user would upload a file via the controller. The controller also allows the users to specify their parameters. Using the function from the model, the controller would then retrieve the data needed from database and carry out necessary analysis. The results of the analysis would then be returned to the View and then the users.

Flask

We build this application using Flask, which is a micro web framework written in python.
We chose flask given the time constraints. Flask provided us with flexibility needed for such a task by not only accelerating development of simple web applications but also providing us with the required functionality. Moreover, it uses Jinja2, a full-featured template engine. This was useful as it is python 3 compatible, allowing python-like expressions in HTML documents. The main libraries used in Flask include: SQLALchemy, flask-WTForms and render_template.

HTML/CSS

```
.
├── Database
│   ├── Database_query.ipynb
│   ├── Database_query_II.ipynb
│   ├── Documentation.txt
│   ├── Images
│   ├── Protocol\ IronFox\ MK\ I.ipynb
│   ├── README.md
│   ├── __init__.py
│   ├── __pycache__
│   ├── db_setup.py
│   ├── kinase_database.db
│   ├── kinase_declarative.py
│   ├── kinase_functions.py
│   ├── kinase_importer.py
│   ├── new_clean_human_kinase_substrates.csv
│   └── schema_for_kinase_db\ -\ actual.pdf
├── Dockerfile
├── __pycache__
│   ├── app.cpython-37.pyc
│   ├── forms.cpython-37.pyc
│   └── user_data_input_parameters.cpython-37.pyc
├── app.py
├── config.py
├── forms.py
├── instance
│   └── Data_Upload
├── kinase_database.db
├── requirements.txt
├── static
│   ├── JsStyle.js
│   ├── help_images
│   └── main.css
├── templates
│   ├── Data_Upload.html
│   ├── GB_Test.html
│   ├── HumanKinases.html
│   ├── Individual_inhibitor.html
│   ├── Individual_kinase.html
│   ├── Phosphosite.html
│   ├── about.html
│   ├── advancedSearch.html
│   ├── data_analysis_results.html
│   ├── data_parameter.html
│   ├── help.html
│   ├── home.html
│   ├── inhibitors.html
│   ├── layout.html
│   ├── results_kinases.html
│   ├── results_phosphosite.html
│   └── results_phosphosite_location.html
└── user_data_input_parameters.py
```

Figure 5. Application layout. The Figure shows our application layout.

The basis of our HTML code was grounded in our approach to provide both a professional and functional website. A number of HTML tags were applied to define the component of each page**.** We were able to utilise render templates in order to simplify the continuous addition of content. We initiated a layout.html file, from which we were able to inherit a base template. This base template was essentially a bootstrap style navigation bar. This was corroborated by a main.css file within our static as shown in Figure 5, which laid out our web application stylistically. Our templates would call into the static folder to enable our own custom CSS style sheets

We opted to display information in a user-friendly manner, namely in tables. However, we implemented bootstrapping to make our tables, forms and buttons more responsible. Bootstrap Is an open source CSS framework for building web applications. This was utilised due to the ease as which content could be added, without the need to manually initiate CSS styles for every component displayed in our website. Also, we decided to implement Bootstrap in other key areas within our

website. One key area was displaying results information. Instead of displaying all the information all on one page, we decided to display results using navigation tabs. This has the added benefit of being visually pleasing. Bootstrap was used due to time constraints within this project.

Application Structure

As Shown in Figure 5, we decided to layout our application in an intuitive way. This included having a forms and app file, alongside both a template and static folder. The app.py contained our different routes for our website. This imports a number of modules needed to run our application. The Database is the module containing the functions to query the database as well as the codes initialising the tables and importing the datasets into the database. The config.py file sets the configurations needed for the application whereas the, requriments.txt states the necessary Python packages version for the app to run. We decided against the use of blueprints as this was a smaller application. We wanted to create a uniform website in which different routes shared the general layout and styles. In addition, debugging issues were far easier due to the uniform structure. The user_data_input_parameters.py file contains the script containing functions that carry out an analysis on the data from the file uploaded by the user, and then outputs html of visualisation of the data.

Running PhosphoView

You can run PhosphoView both locally and on web a server. To run on AWS, please visit the website below

http://phosphoview-docker.xfqyhkzn9j.us-east-2.elasticbeanstalk.com

However, if you intend to run our website locally. Setup the repository create a virtual environment and install the libraries;

```
Git clone https://github.com/MO105/DreamTeam

pip install -r requirements.txt

cd DreamTeam/app/
```

Windows	Linux
`set FLASK_APP=app.py` `flask run`	`export FLASK_APP=app.py` `flask run`

Application runs on `localhost:5000`

Data collection

Kinase, protein names and aliases

The list of human kinases were compiled from a Universal Protein Resource (UniProt) page (at https://www.uniprot.org/). In general, for the sake of simplicity, we have omitted any isoforms and pseudogenes.

The protein names and uniprot entry name for the kinases as well as other synonyms for the genes were then retrieved programmatically from UniProt Knowledgebase (UniProtKB) using the library bioservices, available through python3.7. UniProtKB has been expertly curated to ensure a high level of reliability.

Uniprot has been known to be a secure site, providing free and comprehensive information on protein sequences and functional annotation. Each entry annotated has at least a degree of evidence to it. As such, the site is deemed a trustable source of information for biological data. Moreover, UniProt is updated and distributed every three weeks, making the results to be highly trustable.

Inhibitor

All of the data for the inhibitors were collected from Protein Kinase Inhibitor Database (PKID)(Carles et al., 2018) which currently contains 218 kinase inhibitors, which includes FDA approved inhibitors and also inhibitors that are currently in clinical trials . The latest update on the website was done on 11/12/2019. In the research article, Carles et al give a detailed analysis of this database. All of the kinase inhibitors have an official international nonproprietary name (INN) and are in a clinical developmental phase. PKIDB contains information about each kinase inhibitor such as its molecular weight, target, smiles, inchikey, as well as images. Also, it contains links to other databases such as pubchem and chembl. Chembl ID was extracted from these links. All of this data was extracted by web scraping using Beautiful soup (pkidb.py) and stored in a file called 'pkidb.csv'. The targets in this csv were then compared with our kinase names and all the targets that did not match were removed. Inhibitors that did not contain a target were also removed. This left us with 183 inhibitors stored in a file called ' Complete_inhibitor.csv' which is used in the web application.

Phosphosites

The phosphorylation data was extracted from 'PhosphoSitePlus'. We used this source as it had relevant information regarding kinase and substrate phosphorylation. The dataset we chose from the website was the 'Kinase_Substrate_Dataset.gz' dataset. However, a substantial amount of data wrangling was required in order to clean the data. Once imported into a Pandas data frame, we kept only relevant information relating to only humans. This included features such as; gene name, kinase, kinase accession, substrate, substrate

accession, substrate gene name and sites of phosphorylation modified residue.The phosphorylation data had two primary uses.
Firstly, we return the user phosphorylation sites for a particular substrate, and we use it for the phosphoproteomic data analysis section. However, In order for the dataset to be used as part of the phosphoproteomic data analysis section, further data wrangling was needed. We sought out to append entry names for both kinases and substrate from Uniport. Thus, we used the Python package Bio Services, in order to access Uniport API. We were able to create a function, from which entry names were able to be retrieved when the user inputs uniport IDs. However, 74 entries in our dataset did not return entry names due to errors in naming. Therefore, we used regular expressions which will be able to find and replace those entries in our data frame. This output was named 'new_clean_human_kinase_substrates.csv' and this data was then inputted into our database.

Subcellular Locations of Kinases

Subcellular locations of kinases were retrieved using QuickGO REST API. As there were several sources of evidence for each kinase, there were often duplicates in the data, these were removed from the final list of subcellular locations.

Genomic locations of phosphosites

Finding the genomic location of phosphosites was done in two parts. Firstly, chromosomal position (chromosome, karyotype band and strand direction), were obtained using the Ensemble Biomart search tool. The user-friendly interface made it easy to search within a human database and select to retrieve only the information that was required. This search was further filtered to only search for genes from the uploaded file of Uniprot accession numbers. The file of Uniprot accession numbers was created by taking the column of SUB_ACC_ID from (new_clean_human_kinase.csv).

The genomic coordinates of phosphosites were obtained using the API from EBI (European Bioinformatics Institute) which extracted information from Uniprot. The URLs were generated by using the format function, in python, to add the accession number of each substrate from the list used previously. Using requests.get() to obtain the information from each url and then re.findall() to extract information only on modified residues. From this, the genomic coordinates were extracted using regular expressions and input into a pandas data frame with corresponding substrate accession number. The total number of phosphosites collected in this way was 23,802.

To create the final phosphosite genomic location data frame to be used in the applications database, information from the above data frames had to be appropriately joined and merged. When merging the phosphosite data from uniprot to the phosphosites in the new_clean_human_kinase.csv, which contains 10,981 phosphosites, substrate accession number and phosphosite position were used for the merge. This resulted in a smaller data frame of 7,267 phosphosites. The reason for the data loss was partly due to uniprot containing phosphosites which don't have a known kinase associated so were not appropriate for the final data frame. Also, the phosphosite data was collected from another source (PhosphoSitePlus) which resulted in some differences in phosphosite positions and therefore didn't match the uniprot phosphosite positions exactly. As a result the phosphosites which don't match exactly between data frames are not included.

Database

Database design

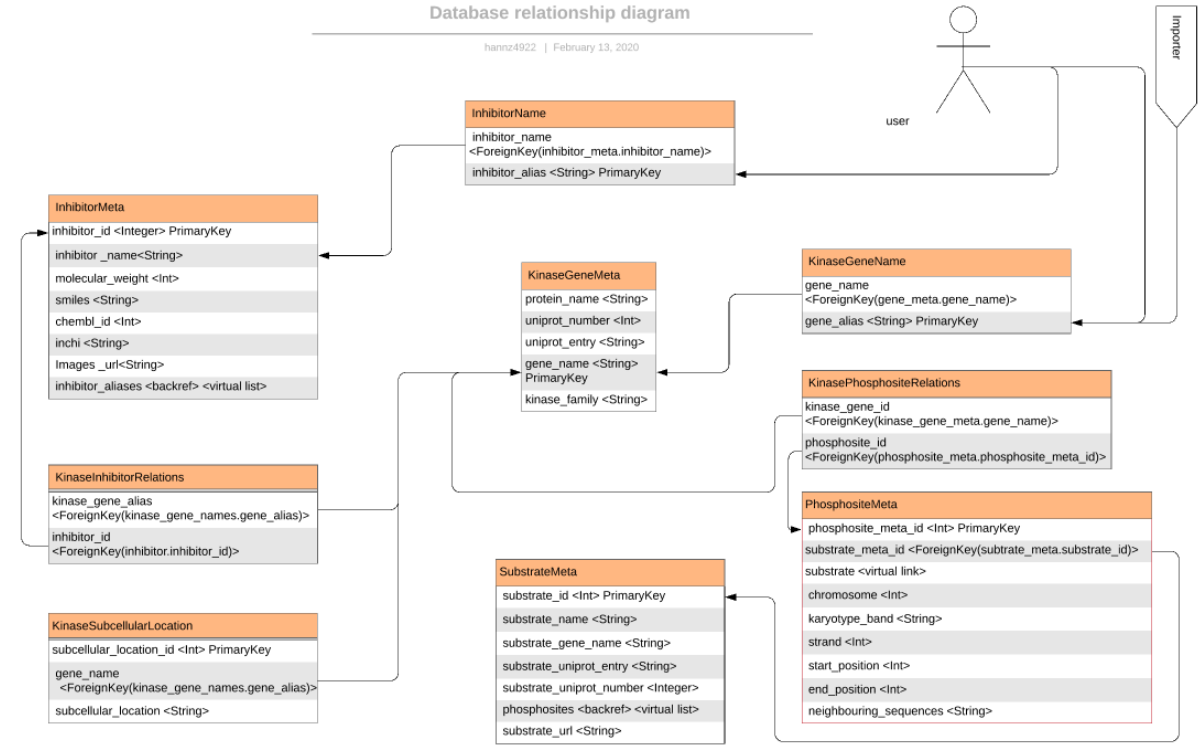

Figure 6. The figure shows the schema for the database.

The database is designed with the intent that users will look up a gene. We understand that a lot of genes have several aliases as different laboratories have found the same gene without realizing that it has been annotated before. Therefore, we have a KinaseGeneName table which align each alias with its respective preferred gene known in Uniprot.

Each entry in KinaseGeneName would be connected to its respective entry in KinaseGeneMeta where the details such as the protein name, the uniprot entry name and the uniprot number are stored. This way, by joining KinaseGeneMeta and KinaseGeneName, we can allow users to search for an entry via 4 different ways: gene name, gene alias, uniprot number and uniprot entry name.

Given that a phosphosite is specific to a substrate and a kinase, it might be more intuitive to combine the metadata for phosphosites in the same table as a substrate. However, this will make it a rather large table. As substrates also have their own metadata, it became clear that substrate metadata table should be on its own. A kinase acts on a phosphosite on a substrate. As such, it is decided that kinase table should be indirectly linked to phosphosites via a relationship table. This is because there is a many-to-many relationship between kinase and phosphosite.

Similarly, there are also many-to-many relationships between kinase and inhibitor, and between kinase and subcellular location. Thus, a relationship table was drawn between kinase and phosphosites as well as between kinase and inhibitor. No relationship table was drawn between subcellular location and kinase because the trade-off between the time saved from using a relationship table and hassle of creating it is not worth it.

Each inhibitor is also expected to have several aliases. Therefore, a table has been drawn up to allocate each inhibitor alias to its respective common name. The inhibitor name table is then connected to the inhibitor metadata table.

There is a one-to-many relationship between substrate and phosphosites as each phosphosite belongs to a substrate. Therefore, a foreign key and a backref pointing towards the Substrate table from the PhosphositeMeta table is sufficient.

The relationship between the KinaseGeneName and KinaseGeneMeta is a many-to-one relationship. Thus, a foreign key is enough.

Database creation and querying

For the relatively small scale of the project, SQLAlchemy is chosen with sqlite. This means the database created will be on a local machine rather than on web-based server. This has multiple advantages over the use of other database languages like MySQL. One of them being that SQLAlchemy is one of the object relational mapping tools that allows users to minimise the necessity for tedious and error-prone SQL statements.

SQLAlchemy is an object-relational mapping tool also allows the user to employ object-oriented programming methods to manipulate and query data from database. Python 3.7 was therefore used for the purpose of this project because of the versatility as well as the accessibility for beginners. On top of that, sqlalchemy is already a library that can be accessed easily through Python 3.7. Web-based SQL also has a limit on how much data the user can store and importing the data to the web-based server is normally unencrypted.

PhosphoView Features

Kinases

The features for kinases which we are going to discuss are in three different pages that are linked. They will be referred to as the search page, results page, and final kinase page. The search page is where the user searches for a kinase. The results page consists of a table of information filtered by the user's input. The final kinase page consists of all the information about a kinase that the user clicks on in the results page.

Search bar

The search bar feature for kinase allows the user to search by gene or kinase name including gene aliases. The user input does not have to be a full match with our gene or kinase name as long as it is a part of a valid name. The application will output a table of results if the user input is found in a valid gene or kinase name. If the user input is not found in our database as a kinase name a red flash bar will appear at the top of the page letting the user know that it could not be found.

Links to final kinase page, uniprot, substrate page, and help page

The results page consists of a table that outputs the gene name,gene aliases, protein name, and uniprot id. The gene names are hyperlinked so if a user wants to find out more about a specific kinase then they can click on the gene name for that kinase and this will take them to the final kinase page for that kinase.

The final page has a link to Uniprot via the uniprot number. If a user wants more information about a certain kinase, then they can click on the uniprot number and this will take them to the uniprot page for that uniprot number.

The final page consists of background information, phosphorylation site, and protein browser. Under the phosphorylation sites tab, all of the substrates are hyperlinked. If a user clicks on the substrate it will take them to a substrate page with information about that substrate.

The search page has a link at the bottom of the search button to the help page. This provides the user guidance on how to use the search bar or any other function they might want to know about.

Protein Browser

We decided to make our website more interactive and user -friendly by embedding the Biojs library, ProtVista. It is a Protein Annotation tool which allows for graphical visualisation of a number of features using the UniProt database.

Download and email button

The results page consists of a table of information such as the gene name, gene aliases, protein name, and uniprot Id which can be downloaded as a csv. The blue download button can be found on the top-right of the page. The results page also has a green email button next to the download button. Once clicked, mailwill open and the message of the email will have the url of the search page and the subject of the email will be “List of Kinases”.

The final kinase page also has the email button on the top-right of the page which once clicked will open mail. The message of the email will be the url of the final kinase page and the subject of the email will be the name of the kinase.

Inhibitors.

The features for inhibitors which we are going to discuss are in two pages which will be referred to as the search page and the final inhibitor page. The search page can be found by clicking on the “Inhibitors” in the navigation bar. The search page consists of a table of all the inhibitors and a search bar. The final inhibitor page is a page with information about an inhibitor which the user chooses.

Search bar

The search bar feature in the search page allows the user to search by inhibitor names, chembl id, or target name. The table of inhibitors condense with the relevant information as the user types in the search bar. This allows easy filtering for the information they want from a long table.

Links to final inhibitor page, chembl, and kinase page and final kinase page

The table in the search page is linked to the final inhibitor page by the inhibitor column. All of the inhibitors are linked to their own final inhibitor page. If a user wants more information about a particular inhibitor they can click on the inhibitor and this will take them to a final inhibitor page about that inhibitor. The final inhibitor page has many information about the inhibitor including chembl id which when clicked will take the user to the chembl website.
The targets in the search page table are linked to their final kinase page. When a user clicks on a specific target it will take them to a page for that kinase.

Download and email button

The search page has a blue download button on the top-right of the page. This lets the user download the whole table as a csv. It also has a green email button next to the download button which when clicked will open mailand the message will contain the url of the search page. The subject of the email will be “list of inhibitors”.
page. The final inhibitor page also has an email button at the top right which opens mail and puts the url of the final inhibitor page in the message section.

Copy to clipboard

The final inhibitor page has a copy feature which lets the user copy the smiles or inchikey on to their clipboard by clicking on the “Copy” text next to them. Smiles and inchikey are specific to an inhibitor therefore it is important for the user to get the correct information. This feature lets the user copy the exact smiles or inchikey onto their clipboard without having to do it manually.

Browsing phosphosites

Another feature of Phosphoview is that the website allows the user to look into genomic locations of substrates and more specifically the phosphosites. If the user wants to look at a specific genomic location they can search using the chromosome and karyotype band options which will take them to a page where they are shown all substrates and phosphosites at that location. Alternatively, if the user is looking to find out the genomic location of a phosphosite they are able to search by the substrate gene name which will show then all phosphosites present on that substrate only.

The result of the phosphosite search also shows the user information about the neighbouring amino acids of the phosphosites present. Furthermore, the user is provided with a link to the UCSC genome browser for that particular substrate. The genome browser has been configured in a way to also include a Uniprot track showing modified residues where phosphosites are shown in yellow. This allows the user to easily visualise the genomic location of a particular phosphosite in relation to the rest of the protein e.g whether the target phosphosite is close to the start or the end of the gene sequence.

Help Page

To enhance the user experience, we have included a help page on the website. This page contains a tab separated table where each tab contains relevant information required for the user to understand and navigate around a particular page of the website. It provides an overview of how the different search functions work. The help page includes screenshots of where to find links to related pages within the website and also to more information on external websites. A comprehensive guide for uploading and interpreting results of the data analysis function is also provided with further link to full software documentation. Additional functions such as how to download, email and copy text is also explained. Links to the help page are provided on various other pages of the website, a further development would be to have links from particular pages go straight to the corresponding tab in the help page table.

Data Upload – Phosphoproteomic Analysis Tool

The phosphoproteomic analysis tool is designed to allow users to upload quantitative data obtained from LC-MS/MS phosphoproteomics analysis. Users can obtain a visual overview of the phosphosite-kinase relationships in the sample as well as estimations of relative kinase activity for every human kinase in the sample.

Data Analysis – Overview

The aim of the data analysis part of the script was to perform operations to extract and clean user data, import the corresponding kinases from the PhosphoView database and calculate the relative kinase activity of the dataset. The output obtained allows visualisations to be produced by other functions in the script.

Python 3.7 was used to write the script for the data analysis. The script primarily utilises libraries such as Pandas and NumPy to handle the large data sets of the files uploaded by the user and to carry out statistical operations, such as taking the log of different values. Additionally, the statistics module SciPy was used to convert the Z-Scores obtained by the kinase-substrate enrichment analysis to p-values.

Checking and Filtering User Data

The data must be uploaded as in TSV format and must have at least 5 columns, in the following order:

- Substrate: The gene or protein name of the protein (UniProtKB entry names and gene names accepted) and phosphorylation site. The phosphorylation site must include a residue type and the numeric location. For example, "1A24_HUMAN(S359)" or "AAAS(S495)".
- Control Mean: The mean phosphopeptide MS1 Peak area in the untreated sample
- Inhibitor Mean: The mean phosphopeptide MS1 peak area in the treatment sample
- Fold Change: The control mean divided by the inhibitor mean.
- P-value: the statistical significance of the fold change

The file can also have 2 extra columns for coefficients of variance for replicates in the control and treated samples, in the following order:

- Control CV
- Inhibitor CV

If there are more than 7 columns in the file, only the first 7 columns will be taken. Therefore, if data for more than one inhibitor is present in the dataset, the analysis will be carried out for the first inhibitor only. If only the 5 compulsory columns in the dataset are present, two extra columns coefficients of variance columns will be added. These coefficients will be set at 0, as it is assumed that if the columns are not included in the original file, then the coefficients of variance will be ignored for the data analysis.

After the file has been uploaded, the user can specify the parameters to be used in the analysis. These include:

- P-Value (p-val): the significance level threshold for the visualisations
- Fold Change (FC): The fold change threshold for significance
- Coefficients of Variance (CV): Filter by coefficients of variance of both treatment and control. Coefficients of variance usually fall between 0 and 2, with a CV < 1 being

considered low variance, and a CV > 1 being considered high variance. As most CVs will fall between these values, the maximum CV the user can choose is 3.

- Minimum Number of Substrates (Sub): the minimum number of substrates phosphorylated by a particular kinase.

Rows with empty columns substrate, control mean, inhibitor mean, fold change and p-value are removed from the analysis. Additionally, if the phosphorylation site is missing for a protein, it is not included in the analysis. Likewise, if the phosphorylation site is stated to be a methionine, the protein is removed from the analysis. While methionine residues are thought to be associated with regulating protein activity, they are oxidised and not phosphorylated (Veredas et al., 2017).

The -Log10 of the p-values are then taken, as well as the Log2 of the fold change. Any fold changes in the data that are infinite because of a 0 in the control column are converted to 0 and any fold changes in the data which are infinite due to logarithm of 0 to the base 2 are converted to 0. The protein and phosphorylation site of each row in the dataset are matched to the corresponding kinases that phosphorylate at that phosphosite, from the PhosphoView database. The final kinase-substrate data is then filtered based on the coefficient of variance as specified by the user.

Kinase Substrate Enrichment Calculations

Kinase Substrate Enrichment Analysis (KSEA) is used to find the relative kinase activity of the dataset. KSEA estimates based on the collective phosphorylation changes of a kinase's identified substrates. Whilst there are 3 main methods of KSEA calculation (Casado et al., 2013; Wiredja et al., 2017; Wirbel et al., 2018), the calculation used in this tool is as follows:

$$\frac{(mS-mP)\sqrt{m}}{\delta}$$

Where:

- mS = the mean fold change of a substrate set of a kinase
- mP = the mean fold change of the entire data set
- Delta ()= the standard deviation of the fold change of the dataset
- m = the number of substrates in a substrate set for a kinase

A negative z-score suggests that that the substrates of a kinase are generally dephosphorylated with the treatment group, which would suggest decreased activity and thus downregulation in the treatment sample, whilst a positive z-score suggests the opposite is true (Wiredja et al., 2017).

Data Summary and Visualisations:

After data analysis three visualisations are generated by 3 separate functions:

1. Summary of Proteins in Sample: Shows all the data points in the sample after the empty rows and substrates without valid phosphosites are sorted out of the data set. Data points shown in green are significantly more abundant in the treatment sample compared to the control. On the other hand, substrates with a lower abundance in the sample compared to the control are shown in red.
2. Kinase Activity Summary: Each data point corresponds to a kinase and the corresponding substrate. Data points coloured green are significantly upregulated in the treatment sample compared to the control. Data points coloured red are significantly downregulated in the treatment sample compared to the control.
3. Enrichment Plot: Shows the Ratio of Enrichment for each kinase found in the sample. Kinases that have a negative enrichment have a lower enrichment compared to the whole dataset, and therefore have a lower relative kinase activity compared to other kinases. Kinases that have a positive enrichment have a higher enrichment compared to the whole dataset, and therefore have a higher relative kinase activity compared to other kinases.

The interactive visualisation library Bokeh was used for all three visualisations, as it allows the visualisations to be embedded easily into the results page as well as the easy implementation of extra design features. For example, each point in the plot can be hovered over to see the data that corresponds to each point. The user can also move around the plot, zoom to view specific points more clearly and save the plot as a PNG image. The thresholds used within the plots for p-value, fold change significance and the minimum number of substrate sets (KSEA enrichment) plots are specified by the user. In addition to the visualisations, the HTML of the KSEA calculations data table was produced by a fourth function.

Discussion: Limitation and Future Developments

The phosphosite data loss that occurs when creating the final phosphosite genomic location data frame used in the database has limitations. The loss of phosphosites with no known kinase means that the user isn't able to explore the complete phosphosite landscape of a protein however the software aims to provide the user with a complete understanding of interactions between kinases and phosphosites therefore the lack of a known interaction would made these entries redundant for our purposes and recovery of this data was not undertaken. The phosphosite data loss that occurs due to a mismatch of phosphosite position information between the data sources used (PhosPhositePlus and Uniprot) also limits the application. A smaller dataset impacts the data upload results as well as not providing the user with the full compilation of known phosphosite-kinase interactions. As this issue was only fully understood around halfway through the project the decision was made to not go back and try to recover lost data.

In regard to the phosphoproteomics analysis tool, whilst it has some flexibility with regards to the dataset uploaded by the user, such as allowing CV columns to not be present, more flexibility could be implemented. For example, the tool could be updated to enable files containing data from multiple treatment types to be uploaded and analysed, and comparisons between those treatment types to be made. Another limitation of the data upload is that currently no specific error message is shown to the user if they upload a file with less than 5 columns or a file with the incorrect column types. It would be useful for the

user to see their file type is incorrect before any analysis is attempted on the content of the input file. Additionally, due to the multiple testing carried out by the KSEA, a p-value correction such as the Benjamini-Hochberg method would have allowed for controlling false discovery rate. However, when carried out on sample data, the Benjamini-Hochberg method produced very few, if any, significant results. This is possibly due to the loss of data caused by merging from different sources when constructing the database. Also, a check for correct fold changes based on the control and inhibitor columns would be useful to ensure that the analysis is correct.

As well as the issues with carrying out a p-value correction, the loss of data causes issues when outputting results from the data analyses. Sample datasets used to test the tool found that kinases that would usually be significantly downregulated by a particular inhibitor were often not significantly downregulated or shown to be upregulated. This could be due to the loss of nearly ⅓ of the potential phosphosite locations. There is also occasionally an issue with the figure key sometimes overlapping with the fold change threshold lines in the volcano plots.

Future developments

The phosphosite result page could be further developed by recovering the phosphosites without a known kinase and incorporate this information in a separate feature of the website so the information would be available for the user if required. Also, data mining from curated sources only, such as Uniprot, and from multiple sources to compile results into a dataset which would be a consensus of multiple reviewed sources would improve the websites.

The phosphoproteomic analysis tool could be further developed in a number of ways. Firstly, as there are several different kinase activity algorithms published in peer-reviewed journals, and two additional versions of the KSEA calculation, greater versatility could be added by allowing the user to choose which measure of kinase activity they would like to use. Additional measures other than KSEA could also be implemented, such as kinase activity ranking using phosphoproteomics data (KARP) (Wilkes et al., 2017),, or the IKAP machine learning algorithm (Mischnik et al., 2015). Further visualisations could also be used, such as kinase-substrate networks of phosphorylation which could potentially show interactions between downregulated and upregulated kinases in the treatment sample compared to the control sample. Also, with the addition of a comparative analysis of different treatment types as mentioned above, visualisations such as a heatmap showing the regulation levels of different kinases under different treatments could be implemented. choose between different algorithms when calculating relative kinase activity.

To further improve the user experience, we would include more links to take the user to relevant pages within the website and to corresponding sections in the help page rather than the first tab.

References

Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M. and Martin, M.J., 2005. The universal protein resource (UniProt). *Nucleic acids research*, *33*(suppl_1), pp.D154-D159.

Binns, D., Dimmer, E., Huntley, R., Barrell, D., O'Donovan, C., Apweiler, R., 2009. QuickGO: a web-based tool for Gene Ontology searching. Bioinformatics 25, 3045–3046. https://doi.org/10.1093/bioinformatics/btp536

Bokeh Development Team (2018). Bokeh: Python library for interactive visualization. URL http://www.bokeh.pydata.org.

Carles, F., Bourg, S., Meyer, C. and Bonnet, P. (2018). PKIDB: A Curated, Annotated and Updated Database of Protein Kinase Inhibitors in Clinical Trials. *Molecules*, 23(4), p.908.

Casado, P., Rodriguez-Prados, J.-C., Cosulich, S.C., Guichard, S., Vanhaesebroeck, B., Joel, S., Cutillas, P.R., 2013. Kinase-Substrate Enrichment Analysis Provides Insights into the Heterogeneity of Signaling Pathway Activation in Leukemia Cells. Sci. Signal. 6, rs6. doi:/10.1126/scisignal.2003573

Ensembl.org. (2020). [online] Available at: https://www.ensembl.org/biomart/martview/ecafbcbaa89563bb4025cc5a4a69d7b9

McKinney, W., 2010. Data Structures for Statistical Computing in Python, in: Walt, S. van der, Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference. pp. 51–56.

Mischnik, M., Sacco, F., Cox, J., Schneider, H.-C., Schäfer, M., Hendlich, M., Crowther, D., Mann, M., Klabunde, T., 2015. IKAP: A heuristic framework for inference of kinase activities from Phosphoproteomics data. Bioinformatics 32, 424–431. doi:10.1093/bioinformatics/btv699

UniProt (2020). Available at: https://www.uniprot.org/ .*UniProt* (2020). .

The UniProt Consortium **UniProt: a worldwide hub of protein knowledge**
Nucleic Acids Res. 47: D506-515 (2019)

Veredas, F.J., Cantón, F.R., Aledo, J.C., 2017. Methionine residues around phosphorylation sites are preferentially oxidized in vivo under stress conditions. Scientific Reports 7, 40403. https://doi.org/10.1038/srep40403

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods. doi:10.1038/s41592-019-0686-2

Walt, S. van der, Colbert, S.C., Varoquaux, G., 2011. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering 13, 22–30. doi:10.1109/MCSE.2011.37

Watkins, X., Garcia, L. J., Pundir, S., Martin, M. J., & UniProt Consortium (2017). ProtVista: visualization of protein sequence annotations. *Bioinformatics (Oxford, England)*, *33*(13), 2040–2041. doi:10.1093/bioinformatics/btx120

Wilkes, E.H., Casado, P., Rajeeve, V., Cutillas, P.R., 2017. Kinase activity ranking using phosphoproteomics data (KARP) quantifies the contribution of protein kinases to the regulation of cell viability. Mol Cell Proteomics 16, 1694. doi:10.1074/mcp.O116.064360

Wirbel, J., Cutillas, P., Saez-Rodriguez, J., 2018. Phosphoproteomics-Based Profiling of Kinase Activities in Cancer Cells, in: von Stechow, L. (Ed.), Cancer Systems Biology: Methods and Protocols. Springer New York, New York, NY, pp. 103–132. doi:10.1007/978-1-4939-7493-1_6

Wiredja, D.D., Koyutürk, M., Chance, M.R., 2017. The KSEA App: a web-based tool for kinase activity inference from quantitative phosphoproteomics. Bioinformatics 33, 3489–3491. doi:10.1093/bioinformatics/btx415