Data mining

To retrieve the data, we had four primary sources:

* Uniprot: a database of proteins, containing a variety of functional and structural information
* European Bioinformatics Institute which has also contains a variety of information on many proteins in an easily parsable XML format
* Ensembl which provides genomic information on a number of proteins and their transcripts
* PhosphoSitePlus: a database of post-translational modifications in human, mouse and rat proteins

The datamining was completed via python scripts and the various APIs available on the respective sources. The data was incorporated into a number of csv files, which could then be used to populate our database.

Kinase data

For our database we required a variety of information on human kinases. We retrieved a list of human kinases from the uniport database (<https://www.uniprot.org/docs/pkinfam>) and used this list to decide which proteins we needed to obtain information for. We then use the uniport API (<https://www.uniprot.org/help/api>) as well as the European Bioinformatics Institute protein data bank (<https://www.ebi.ac.uk/pdbe/node/1>). With EBI, we were able to get protein information in an XML format which could then be parse using the BeautifulSoup package in python to retrieve the relevant information. This information included the gene and protein names for each kinase, the domains and positions, the phosphosites on these kinases as well as the protein sequence for each kinase.

Substrate and phosphosite data

Substrate and phosphosite data was taken from PhosphoSite Plus. More specifically, the kinase-substrate dataset (<https://www.phosphosite.org/staticDownloads>). The data was then filtered to only include phosphorylation of human proteins. This data provided us with a list of substrates and phosphosites as well as information about the kinase, the kinase accessions, the substrates, the substrates accessions and locations of phosphorylation.

We then filtered the relevant data using python and made a number of .’csv’ tables according to our established data schema. The generation of tables was primarily done through the ‘pandas’ package in python.

SQLite

For the generation of our database and the connecting of our database to Flask we chose to use SQLite. The primary reason for this choice over other options such as MySQL was primarily due to SQLite not requiring a client-server engine. This simplified our architecture and allowed us to easily integrate the database into our website and deploy it through AWS Elastic Beanstalk. For our relatively small database, SQLite serves us adequately and allows for fast querying and loading of data.

Genome Browser

To allow users to browse phosphosites by their genomic location, we incorporated a custom embedding of the NCBI sequence viewer. Our genome browser allows users to select a chromosome to view and the viewer will display the genomic sequence of said chromosome with markers placed at phosphosites within the chromosome. The markers are named with the accession of the phosphosite along with the amino acid residue phosphorylated and the position of said amino acid residue within the protein. Also available on our sequence viewer are the six reading frames for the genome.

Data limitations

Our data is somewhat limited in the sense that there is some vital information missing. For example, for the phophosites of a given protein, there are many cases where we have no information what kinases are responsible for this phosphorylation. It is possible that this data can be found from other sources, or potentially from further experiments in the field. Also due to the time constraints in developing this webapp, there is also information missing about kinases and substrates, such as their functions or if they are associated with any diseases.

Our inhibitor data is also limited. We are missing many inhibitors again due to time constraints and also the sources we used. If provided further time to develop we would like to increase the size of our inhibitor database as well as provide further information about these inhibitors.

Genome Browser limitations

While our genome browser provides valuable functionality, it also can be fairly slow when looking at a chromosome with a large number of phosphosites. This is likely due to the NCBI sequence viewer not being designed to hold a large number of markers. This could potentially be solved by using a different genome browser more suited for our needs or perhaps providing a genome browser for each substrate which would reduce the number of markers in any given genome browser.