Fachhochschul-Masterstudiengang
BIOMEDIZINISCHE INFORMATIK
4232 Hagenberg, Austria

Scansite 4: Introducing multiple enhancements and conditional phosphorylation to a kinase based database search application

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science in Engineering

Eingereicht von

Thomas Bernwinkler, B.Sc.

Betreuer:	Michael B. Yaffe, M.D., Ph.D. Koch Institute for Integrative Cancer Research Massachusetts Institute of Technology Cambridge, MA, USA
Begutachter:	FH-Prof. PD DI Dr. Stephan Dreiseitl

Juni 2017

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

place, date name, signature

Contents

Acknowledgements

I would like to express my gratitude to Michael B. Yaffe and every single member of his lab for providing such a great work environment. I particularly would like to give my thanks all members of the *kinase gang* for testing *Scansite* and providing great feedback and literature, which made it possible to evolve *Scansite* to the application it is today. Moreover, I really appreciated the financial support, the opportunity to set up a new server environment, and last but not least all the input and advice I was given in meetings.

Also the Austrian Marshall Plan Foundation played an essential role in this project. I am extremely grateful for their financial support, which helped me to finance my research at MIT. At this point I would also like to express my gratitude to Christina Huber-Beran, who brilliantly managed any necessary paperwork to apply for the Marshall Plan scholarship. I really did highly appreciate everything she did to make a scholarship application possible on late notice.

I would like to give special thanks also Konstantin Krismer for supporting me with the development of the software. It was a pleasure to work together with him and I was really grateful for any advice or explanation, especially during early development stages. Speaking of support, I really appreciated the assistance of Tanja Grill. Thanks to her I was able to deal with MIT related paperwork almost effortlessly. In addition she really helped me out to find housing possibilities, which felt like a huge achievement at that time.

I would like to express my gratitude also to Stephan Dreiseitl from my home university, who excellently supervised me during my stay in the U.S. and brilliantly supported me to make the best out of this thesis.

Last but not least I would like to thank all my friends, who make the world a better place for me. I am grateful to all my friends in Cambridge, MA for the great time we had together. At the same time I thank all my family and friends in Europe, who stayed in touch despite having six hours time difference.

Kurzfassung

Scansite ist ein Programm zur Untersuchung und Darstellung von Zusammenhängen im Bezug auf die Phosphorilierung von Proteinen. Die Applikation basiert auf der Programmiersprache *Java* und ist über jeden gängigen Web Browser erreichbar. Im Rahmen dieser Arbeit wurden vier Projektziele umgesetzt. Auch wenn jedes Ziel aus individuellen Hintergründen heraus definiert wurde, tragen dennoch alle zur langfristigen Verbesserung des gesamten *Scansite* Projekts bei.

Die erste Zielsetzung bezieht sich auf die Weiterentwicklung der *Scansite* Hauptkomponenten. Sowohl das Verbessern bestehender Funktionalitäten als auch das Einführen von Neuerungen steht hier im Mittelpunkt. Darunter befinden sich Verbesserungen wie etwa neue Eingabeparameter oder erweiterte Filter. All dies führt zu neuen und erweiterten Möglichkeiten, um mittels *Scansite* Zusammenhänge zwischen Kinasen, Proteinen und deren Substraten zu analysieren. Weitere Neuerungen wie die Neugestaltung von Motif Logos helfen zudem dabei, das Programm intuitiver zu gestalten und relevante Inhalte zugänglicher zu machen. Eine herausragende Neuerung bildet die Berücksichtigung von potentiell modifizierten Bindungsstellen. Dies hat wiederum zur Folge, dass durch andere Modifikationen bedingte Phosphorilierungen berücksichtigt werden können. Mit Hilfe von bedingten Reaktionen kann schließlich ein viel breiteres Spektrum an Suchmöglichkeiten angeboten werden.

Eine zweite Zielsetzung konzentriert sich darauf, die künftige Entwicklung von *Scansite* so unkompliziert wie möglich zu gestalten. Dies beinhaltet Punkte wie das Aufsetzen einer Datenbank oder das Vorbereiten einer Serverumgebung. Durch neu implementierte Installationsdateien ist es künftig möglich, eine voll funktionstüchtige *Scansite* Installation mittels weniger, gut dokumentierter Schritte aufzusetzen und der Öffentlichkeit zugänglich zu machen.

Die Schaffung einer vom öffentlich zugänglichen System unabhängigen Test- und Entwicklugsinstanz war ein weiteres Ziel dieser Arbeit. Dies soll einen störungsfreien Betrieb der öffentlichen *Scansite* Instanz während der Entwicklung neuer Komponenten gewährleisten.

Scansite 4 ermöglicht es, Benutzerkonten zu verwalten und spezifische Inhalte anzulegen bzw. abzufragen. Um ein Mindestmaß an Datenschutz zu gewährleisten, war die letzte Zielsetzung dieser Arbeit die Erhöhung der Systemsicherheit. Beispielsweise wurde die verschlüsselte Übertragung von Daten ab *Scansite* 4 mittels HTTPS umgesetzt. Entsprechend werden Anmeldedaten von Benutzern nicht mehr im Klartext übertragen, wodurch die Sicherheit des gebotenen Service drastisch erhöht wird.

Abstract

Scansite is a protein database search tool, that is able to comprehensively map protein kinase substrates. The application is *Java* based and accessible via web browser. This work focuses on four particular aims. Although these goals were set with different purposes in mind, all of them contribute to improving the entire *Scansite* project for the long term.

The first and major aim is defined by improving the *Scansite* core application itself. By enhancing existing features and introducing innovations, such as new input parameters, filters or editing result pages, important new possibilities to run searches for kinase substrates are provided. Additionally, changes such as a motif logo rework make the software more intuitive and ensure a better understanding of relevant motif-related attributes. An innovation with high impact is based on introducing potentially modified sites and thus conditional phosphorylation to *Scansite*, which offers a new range of possibilities for searches.

A second aim focuses on future development and on as effortless software engineering as possible. Thus relevant parts like setting up a database or preparing a software side server environment on a computer have been worked out and nearly fully automated. Consequently, in future it is possible to setup a fully sophisticated *Scansite* server system by executing just a few files.

Another goal takes measures to work independently of the actual *Scansite* server. This is necessary to keep the online system accessible and safe from changes, that are still in developing or testing stage. Innovations, that are not tested properly, could easily break the system and as a result the *Scansite* service would not be available anymore. To avoid overlapping between innovations and the reliable online system, a local development setup has been worked out. By documenting the whole process, also assistance is provided for setting up a local test setup quickly and easily.

As new features such as additional access privileges based on a user login system are available after developing *Scansite* to version 4, user credentials as well as any other private information should be kept safe. As a consequence, the final aim of this thesis concentrates on security features and user management. This, for instance, includes encrypted traffic by introducing HTTPS support. As a result user login data is not transmitted in plain text anymore, which significantly increases the security of the service.

Chapter 1

Introduction

In the past decades, progress in medicine and biology has reached entirely new dimensions. Especially in those fields of research, the amount of data, which is generated each year, is increasing exponentially. Consequently, the amount of information with regards to the human genome and proteome has also reached an outstandingly high level. Moreover, these life sciences have explored many highly important facts in terms of molecular interactions as well as synergies among multiple reactions called pathways. Based on several accomplishments in biology, new ways to treat and partially even cure a number of illnesses could have been developed, which was considered impossible only a few years ago. Even though nowadays science has improved drastically, there are still numerous diseases, that can not be treated well yet. Some of the most famous representatives of this difficult category are various forms of cancer. As this medical issue can be described as uncontrolled growth of cells caused by disorders in inter- as well as intracellular processes, potentially connected proteins and other molecular structures have been examined to determine the cause of any specific ailment. Reactions such as protein-protein and protein-nucleic acid interactions are supposed to regulate most procedures of a cell [10]. Additionally, the mentioned interactions are very often depending on post-translational modifications (PTMs). These differences on a molecular level change the behavior of structures such as proteins as well as their interaction partners which are e.g. cell membranes. Very common modifications are phosphorylation of serine, threonine and tyrosine [15]. Moreover, protein kinases effect control mechanisms of intracellular processes due to phosphorylation of substrates [28]. This kind of reaction however, is very specific and also reversible. Once a fitting protein is found, phosphorylation has to be triggered on a micro-molecular or even atomic tier. Whether phosphorylation is triggered or not depends on a certain attribute called peptide specificity. As this value heavily depends on protein kinases, members of that group have the ability to recognize a specific peptide by its sequence to modify the phosphorylation acceptor site. (Figure 1.1) [7].

1.1 Relevance of Computational Research Support

As mentioned above, the amount of produced data is steadily increasing. Hence people are not able to handle these dimensions of data manually anymore. Amongst

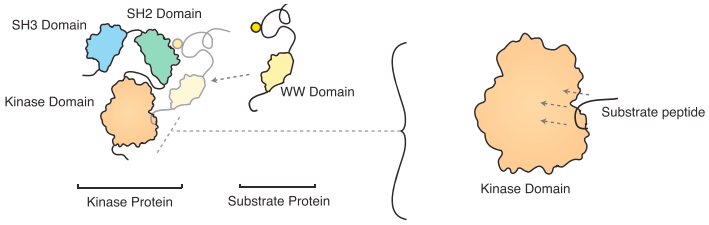


Figure 1.1: An example for both, protein as well as peptide specificity, is depicted to stress differences. Whereas protein specificity depends on a rather macro-molecular structure such as certain protein domains or only single protein 3D structures, which reduce the possibilities of potential reactions on a higher level, a peptide specificity is even more restrictive, and thus only reacts, if a certain binding site or even a required minimum number peptides is available. [7, p. 187-188]

other research topics, this affects the analysis of proteins and molecular pathways. Genomic information in particular increases quickly, which requires new methods to conclude to protein functions and protein-protein interactions [30]. Since *in vitro* and *in vivo* experiments can be very expensive, financially as well as time consuming, reducing both, especially *in vivo* experiments, to a necessary number is often required. As a consequence, one has to come up with a way, that is able to lead to the right direction, before working on the laboratory bench. Computational approaches offer the opportunity to provide necessary information to set a good starting point for further, more detailed research. The virtual attempt is connected to an even greater number of advantages. By using *in silico* systems and various databases, a considerable number of different data sources can be taken into account. Those are often used to gather general information before getting to the actual point of interest. Additionally, by using computational predictions, it is even possible to omit certain experiments, as some could have already been conducted and many results are freely available on public databases. By analyzing and combining data, new conclusions can be drawn which can be used to infer new hypotheses and models. Thus *in silico* approaches are able to potentially replace expensive "wet-lab" experiments or at least provide a baseline for future research.

Together with a growing number of facts in fields multiple of research, reliability and accuracy of computational systems are increasing, which results in an even more promising and versatile support in future. Anyway, one has to consider that those methods are only as good as any underlying data and experiments. It is never guaranteed, that a software application is capable of coming up with the ideal solution. Hence one should be careful while interpreting any results that are provided by an *in silico* method and accept them under reserve.

In modern research, many different software applications are in use and many more are still being or going to be developed. One of those software tools called *Scansite* is a database search application, which is able to predict certain coherences [10, 22, 30]. *Scansite* is a free accessible web based *Java* application, which is able to identify and display various connections of proteins, sequence motifs and phosphorylation acceptor sites as well as protein attributes such as molecular weight, isoelectric point (pI) or amino acid (AA) composition.

1.2 The Aim of *Scansite*

Scansite is a web browser application, that offers the opportunity to comprehensively map protein kinase substrates by computer database scanning. The program uses a number of public databases such as *NCBI*, *SwissProt* or *Ensembl* in order to run search inquiries [10]. The major aim of further developing *Scansite* is to better predict new substrates using kinase motifs, that have been determined experimentally.
Not only motifs, but also various forms of meta data and information about previously mapped sites, mainly provided by *PhosphoSitePlus*, are used to reach this goal. Based on *Scansite* predictions, new mechanisms like regulating the *ETS1* transcriptional oncoprotein by *Src* family kinases can be identified [20]. In a more general view, the ultimate aim for *Scansite* includes the process of scanning the entire proteome of healthy and sick human individuals in order to come up with new coherence on a proteomic level which respectively provides the opportunity to infer a machinery based on genes or even DNA. Both, potentially and ideally, *Scansite* is going to become a key to further research interactions on a molecular level and consequently also a key to develop new drugs.

The software project has evolved many times and from version 2 to later versions the entire front and back end has been transferred to a new software architecture and even to a new runtime environment. Whereas *Scansite* 2 and earlier versions were implemented in C/C++, *Scansite* 3 Obenauer, Cantley, and Yaffe [22] and later, not yet published releases, are based on Java.

1.3 Motivation for Developing *Scansite* 4

The *Scansite* software project was evolved to version 4, as described in following chapters. Major objectives for doing so included four goals. The most important innovation is based on introducing modifications to a local copy of the *UniProtKB* database. This allows to consider conditional phosphorylation triggered by specific kinases. Depending on certain conditions like modifications, a confirmation (three-dimensional structure) of a protein is able to change. Thus hidden parts of the protein come to the surface, which anon can be phosphorylated. As previously mentioned, phosphorylation can trigger various inter- as well as intracellular processes. Consequently, being able to examine conditional modifications makes it possible to define an entirely new baseline for future research. Eventually, introducing these modifications is part of one of the previously mentioned four aims. This first goal is defined as improvement of the *Scansite* core application as well as related software in general.

Other aims are less focused on a user experience. Those cover (i) preparations, automations and implementations to make future development easier as well as making it possible with way less effort, (ii) drastic improvements of the *Scansite* database setup and management to provide a more reliable environment, and (iii) last but not least also taking security measures to keep the application as safe as possible.

1.4 Outline

For the purpose of better understanding *Scansite* and its enhancements, the Background chapter (Chapter 2) explains the terminology and the structure of various data sources and types of data, that are essential for *Scansite* to run. As the application works with different types of data at once to come up with new matches, it is inevitable to know what its underlying data is about. The Background chapter is also supposed to give insights into the basic mechanisms and workflows of *Scansite*. This includes an overview of all currently available features including input parameters, purpose and result output. Additionally, more detailed insights on matches are given. The scoring process, which determines *Scansite* result matches, is an essential part to understand how the application works.

Once, the basics are explained, the Implementations and innovations chapter (Chapter 3) includes the most recent developments of *Scansite*. With these changes, the web application is able to work even more efficiently and also offers new functionality, which makes it easier to get new insights into certain mechanisms. Most of the information, that is retrieved by newly identified mechanisms, is related to cancer research and as a consequence to future cancer treatment. Major introduced features are for instance procedures, that are able to execute a number of tasks automatically to make future development easier. Also improving previously available features was necessary.

The chapter following the Implementations and innovations part (Chapter 4) covers implementations of this work in a user perspective. As a consequence, significance and new possibilities based on improvements are described. In addition further innovations and their advantages are introduced. Chapter 5 discusses the use of innovations, room for improvements and potential future development of the *Scansite* project. This particular part of this work focuses primarily on optimization and mostly realistic development of future features. Chapter 6 summarizes most relevant elements throughout all previous chapters. This Conclusion stresses major developments as well as most useful future changes.

Chapter 2

Background

2.1 The Structure of *Scansite*

As previously mentioned, *Scansite* is a freely available software application which can be accessed using a web browser. The currently in Java implemented version uses a technology stack, which can be split into three major categories. The first one called *Apache Maven* is a build automation tool, which especially was developed for Java support and is also useful for project management as a command line tool [21]. In addition, by using *Apache Maven*, also project configuration and software library management in general are handled. This includes downloading and providing newly required sources during development. A second essential technical component is the *Google Web Toolkit* (GWT). GWT is a free, open source development kit for *asynchronous JavaScript and XML* (AJAX) applications, which defines the major server-client architecture and thus the project structure of *Scansite* [33]. Finally, the database management tool in use is *MySQL*. This type of database and its query language is used on a large number of open source systems, since it is a flexible, lightweight and powerful tool [29]. *Scansite* uses *Maven* to generate a *Java Web Archive* (WAR) file, which is able to be deployed on an *Apache Tomcat* server as web application. Besides, GWT deals with messaging tasks, that are related to user interactions and a *MySQL* database stores all relevant search information.

Next to technological aspects, the general architecture of *Scansite* can also be divided by its major use. As depicted in Figure 2.1, the structure can clearly be split into three parts: (i) The data management component, which is only relevant for administrative tasks such as setting up and maintaining the database, (ii) the web application, which is the core program, a user can access via web browser and (iii) a web service, that offers a REST interface to access *Scansite* computationally, which is especially useful for bulk searches. Furthermore, the second part also contains a login option, which leads to further modules, that anon allow administrative operations like adding new biological sequence motifs. However, this access is limited to a limited number of people, who are responsible for maintenance service. It is not planned to offer a registration mechanism in the near future.

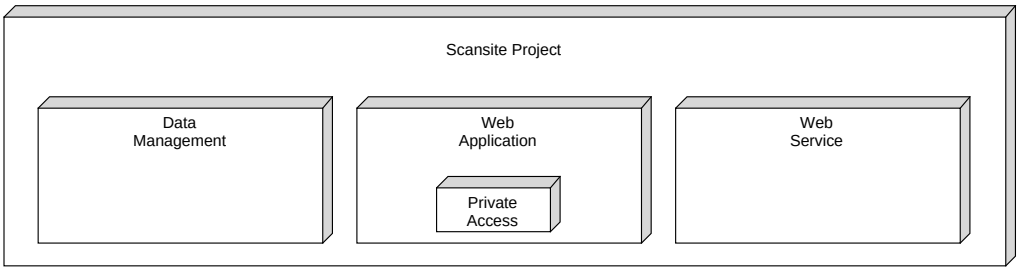


Figure 2.1: Central components are discerned with regards to their primary use. Whereas data management part is used to setup and maintain the server system, the web application is freely available and accessible through the web. Additionally, the actual *Scansite* web program offers a user login, which enables access to administer operations, that anon require certain privileges. A third module is the web service, that provides the opportunity to computationally access *Scansite* through a REST interface. This way it is also able to run bulk searches without constantly going through the form of the website.

2.2 Biological Background

In order to describe features of *Scansite* as well as any procedures running in the background, specific terms regarding biology are introduced. Hereby, the highly relevant process of phosphorylation together with hereof essential components, such as the biological group of kinases and their substrates as well as sequence motifs will be explained. It is necessary to have an idea of these keywords, that build the backbone of *Scansite* to be able, to follow its processes.

2.2.1 Phosphorylation and Kinases

Next to various data sources, *Scansite* heavily depends on sequence motifs, that have been determined experimentally. For generating these valuable motifs, an appropriate experiment includes modification processes like phosphorylation, and molecular structures such as kinases and kinase substrates. Hence these terms are explained in the following context.

Beside acetylation, methylation and sulfation of proteins also phosphorylation is a post-translational modification (PTM). A PTM can be described as a fully translated polypeptide chain, that is modified by adding or removing a molecular group, which consequently transforms a polypeptide chain into a biologically active protein [15]. A (poly)peptide is a short sequence of amino acids (AAs). Hereby the amino acids are connected to each other by a compound called *peptide bond*. The process of phosphorylation becomes especially important in the context of protein kinases. A kinase is an enzyme, which favors transferring a phosphate group. This transfer is a synonym for phosphorylation. The phosphate is usually separated from adenosine triphosphate (ATP) and is transferred to the target molecule. Hereby, the kinase acts as a catalyst, that supports the phosphorylation of the binding site. The phosphate receiving molecule is called substrate. A binding site is defined as a single specific amino acid, that binds a chemical group. In this context such a group is a phosphor

molecule, which anon is provided by ATP molecules. Moreover, in a more specific context, a binding site is also called phosphor acceptor site. Whether a binding site receives a phosphor molecule or not also depends on surrounding AAs and their interactive forces respectively.

To underline the high influence of PTMs in cellular processes, Begley et al. [3] show, that a mechanism called *experimental growth factor receptor (EGFR)*, which is supposed to contribute to a number of human cancer types, contributes more likely to phosphorylation, if there is another prior phosphorylation. In this example mechanism, EGFR is one instance out of a huge number of kinases, that shows the relevance of these modifications in cell signaling processes in general. As EGFR also plays an essential role in regulating cell proliferation, differentiation, migration and apoptosis, essential cell functions can be disabled, just because some reactions do not run the way they are supposed to [3, 32]. Presence or absence of PTMs can be a source of malfunction in cells, which results in intensive biological research. Thus PTMs and molecules like previously described kinases, which lead to various modifications (in this context primarily phosphorylation), are studied more and more intensely.

As these processes are outstandingly relevant in terms of cancer treatment, *Scansite* focuses on predicting phosphorylation sites in proteins. Based on these predictions, researchers are able to obtain new insights and can even come up with new ways of understanding cellular interactions, and hence it is possible to introduce new experiments and potential future treatments with regards to various cancer types. Usually kinases catalyze only phosphorylation of very specific binding sites, that are defined by a particular amino acid, which acts as phosphor acceptor site. Surrounding AAs also have influence on the catalytic function of kinases due to chemical forces. The recognition of these special regions of a potential substrate is expressed in a sequence motif. In the following sections, the term *motif* will always refer to biological sequence motifs.

2.2.2 Sequence Motifs and Scoring Matrices

A motif is defined as Position Specific Scoring Matrix (PSSM). This can be described as a matrix with (dis)favor values for a sequence of amino acids, which in *Scansite* always covers exactly 15 residues on positions $[-7; +7]$. *Scansite* 3 and earlier versions have used all 20 natural amino acids as potential candidates, that can be picked for any of the possible residue slots of a motif. Moreover, there is no limitation to the number of appearances of any AA within a motif, which means that any AA can potentially be found in every position with a relatively high positive value. Only the position in the middle, also referred to as central or zero position, is usually either Serine (S), Threonine (T) or Tyrosine (Y). Those particular three amino acids are most often phosphorylated, compared to all other AAs. Consequently, a residue in the central position is the one, that receives the phosphor group and is also called phosphorylation site, binding site or phosphor acceptor site. As S and T have a very similar chemical structure, motifs can be categorized as (i) those with a Serine/Threonine phosphorylation site, having S or T in the zero position (S/T motifs) and (ii) those with a Y phosphorylation site at the central position (Y motifs) respectively.

Next to the symbols of the 20 natural amino acids, Ehrenberger [9] describes further one letter codes for rare amino acids as well as wild card values, that represent a number of very similar AAs. *Scansite* is able to potentially recognize and use the codes as listed below, although some of the symbols such as super rare amino acids Selenocysteine (U) and Pyrrolysine (O) are very unlikely to appear in a motif and are currently not included in any of the available motifs. However, for *Scansite* searches they are assigned the values of Cysteine (C) or Lysine (K) per default, unless there are specific values defined in future motifs.

- **U:** Selenocysteine
- **O:** Pyrrolysine
- **B:** Wild card for Aspartate or Asparagine
- **Z:** Wild card for Glutamate or Glutamine
- **J:** Wild card for Leucine or Isoleucine
- **X:** Wild card for any residue
- **$ (dollar sign):** Preference for N-terminal sequence
- *** (asterisk):** Preference for C-terminal sequence

Usually a motif can not be determined easily, and as a consequence there is not just a single correct sequence of residues, as biological mechanisms are way too complex. To deal with this complexity, a method, that considers the potential presence of any AA for any residue position, is required. Hence a common approach is comparing intensity or affinity values in a matrix structure. This is applied in *Oriented Peptide Library Screening (OPLS)* experiments, which generate the data for all *Scansite* motifs. OPLS can be described as an *in vitro* procedure, that is able to generate phosphorylation motifs for both, Serine/Threonine kinases as well as Tyrosine kinases, relatively quickly by detecting radioactive material [17].

Using this method, Alexander et al. [1] discovered a number of motifs (like *Cdk1*, *Plk1*, *Aurora A* and *Aurora B*), that are available in important features of *Scansite* such as different searches or displaying information about a motif itself. To develop a better understanding of the most relevant input data used by *Scansite*, and also to be able to interpret the motif representation of the application, it is relevant to describe the process of OPLS.

Peptide library screening is, according to its name, based on given peptide libraries that are provided by *well plates*. All peptides within a library are tagged with biotin, a molecular group, that is relevant in later steps of the procedure [17]. Whereas Alexander et al. [1] used degenerate oriented peptide libraries of a 384-well microtiter plate, it is also able to use bigger (e.g. 1536) well plates to run multiple kinases simultaneously in order to speed up work with different kinases. In this method, a target kinase cleaves the phospho group of from radioactive ATP and phosphorylates appropriate sites of peptides within the library. While phosphorylation reactions are running, the temperature is constantly at 30°C, as most kinases are supposed to work best around that temperature [1]. The next step contains transferring a certain amount of aliquots to an avidin-coated membrane, which binds to the previously mentioned biotin tag, only limited substrates have [17]. Only proteins of the peptide library are tagged with biotin. Thus only peptides and substrates of the library stick

to the membrane. Consequently, noise signals can be reduced to a bare minimum after a number of washing steps. Finally, the dried membrane is exposed to a *"PhosphorImager"*, that is able to identify and quantify the radioactive phospho groups [17]. Those detected groups are the result of phosphorylation catalyzed by a kinase. This is the only way for the radioactive phosphor to be separated from ATP to bind to the peptides, that actually has an impact on results, ignoring randomly binding molecules because of very low reactivity. An example result (Src kinase [3]) of the peptide library screening process in its final state is depicted in Figure 2.2. This shows an image, that has been created based on measures of radioactive signaling on the membrane. Dark spots represent a stronger signal and means high affinity for an AA to be present at a certain position.

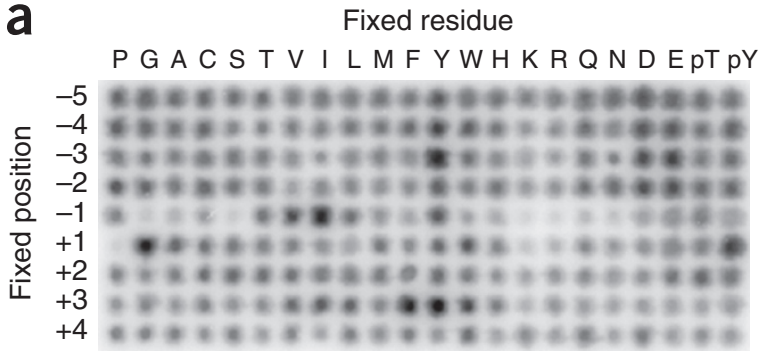


Figure 2.2: An Oriented Peptide Library Searching (OPLS) example substrate motif is shown to demonstrate preferences and disfavor of peptides depending on a kinase of interest. To be more specific, the Src kinase motif [X-X-E/Y-X-I-Y-pY/G-X-Y-X] is depicted. For each position of the site $[-5; +4]$, a signal intensity can be recognized. This value represents an affinity of an amino acid to be in its respective position(s) of the substrate site. Hereby a darker visual expression correlates with an increasing preference. [3, p. 187-188]

Whereas *in vitro* experiments work with intensities, *Scansite* is only able to work with data on which mathematical operations can be applied directly. Hence, the appropriate data structure of the software application uses a comparable method, which is based on numerical representations. Using numbers instead of visual signals highly simplifies processing motifs computationally. All results are transformed into matrices of intensities, that are represented by floating point numbers.

These matrices can be described as *Position Specific Weight Matrices (PSWM)*. By applying a mathematical operation on a PSWM, underlying data is normalized with respect to a predefined background. In this case, every value is divided by a constant number $\frac{1}{n}$ whereby n is the number of possible amino acids, assuming that any amino acid is equally common. Even though this assumption is not correct, in order to reduce complexity, this value is still in use. This way the matrix is converted to a *Position Specific Scoring Matrix (PSSM)*. These PSSMs are the baseline for many calculations in *Scansite* and can be seen as the motifs used by the application. To simplify this matter, Table 2.1 shows most important sections of an artificial *Scansite* motif. A generated version of *CDK1* is listed to point out example preferences.

Table 2.1: Values below describe most relevant sections of a Position Specific Scoring Matrix (PSSM) of the in *Scansite* available *in silico* motif *CDK1*. These PSSMs usually cover all positions from -7 to +7. Next to the other important and highlighted positions, *CDK1* is clearly a Serine/Threonine motif (S/T motif) as these values in the zero position are greater than any other value within the matrix. Another typical appearance in S/T motifs is Proline on positions right after the poshorylated site or at least very close to the +1 position [31]. A visual representation for better understanding is provided in Figure 4.7b. Some less relevant columns of the motif are missing due to limited space.

Position	A	C	D	E	K	L	M	N	P	R	S	T
-7	1.36	0.18	0.63	2.02	1.01	1.78	0.24	0.66	1.99	0.73	1.53	0.52
-6	1.71	0.24	0.66	1.46	1.15	1.12	0.31	0.70	2.26	0.94	1.25	0.70
-5	1.25	0.11	0.87	1.88	0.56	1.46	0.42	0.66	2.26	1.05	1.57	0.42
-4	1.15	0.07	0.84	1.81	0.52	1.53	0.21	0.80	2.06	0.91	2.02	0.77
-3	1.39	0.07	0.52	1.15	0.80	2.79	0.45	0.56	2.20	1.15	1.29	1.01
-2	1.88	0.21	0.63	0.80	0.31	1.85	0.14	0.49	4.18	0.63	1.53	0.70
-1	1.81	0.18	0.59	0.98	0.84	1.99	0.31	0.49	1.71	1.36	1.71	0.94
0	0	0	0	0	0	0	0	0	0	0	21	21
1	0.04	0.07	0.11	0.14	0.11	0.14	0	0.14	19.22	0.07	0.18	0.07
2	1.60	0.07	0.45	1.15	1.22	1.29	0.14	0.63	2.16	1.67	1.25	0.70
3	1.01	0.04	0.31	0.70	3.34	0.84	0.24	0.45	1.71	3.62	1.25	0.70
4	1.88	0.24	0.80	1.15	1.85	1.60	0.31	0.59	2.09	1.64	1.29	0.45
5	1.32	0.28	0.63	0.94	1.74	1.60	0.11	0.56	2.13	2.02	1.71	0.45
6	2.13	0.21	0.52	1.01	1.43	1.71	0.14	0.49	1.99	1.36	1.71	0.63
7	1.25	0.21	0.59	1.12	1.19	1.60	0.21	0.77	2.82	2.06	1.39	0,59

Numerical representations are a decent baseline for calculating scores. However, it is difficult for the human mind to interpret these numbers. On top, it is not easy to guess intensity values of OPLS *PhosphorImager* results. Hence another form of displaying motif data is essential. Next to the numerical server side representation of a motif, *Scansite* is also able to visualize motifs in one of the most common ways, a motif logo, as depicted in Figure 2.3. The "14-3-3 Mode 1" motif of *Scansite* was selected, since it is the default motif in the *Databases and Motifs* sub page of *Scansite*. Whereas the coloring schemes in motif logos often are used to group amino acids with similar attributes by applying the same color, *Scansite* 3 chooses colors that differ too much from each other to recognize similarities. Consequently the coloring does not contain any intuitive information.

2.3 Scoring and Input Data Categories

Scansite includes a non-trivial scoring system, which determines effectively all results. Before going into detail, it is important to know what this value actually represents. A *Scansite* score is defined as likelihood of a peptide to become a substrate of a certain kinase. A 15 residue string of a protein sequence is matched to the motif PSSM, which represents a kinase activity. The value of each AA of the peptide is determined by its amino acid letter and its position within this short sequence, as those values determine a position inside the PSSM. In total a number for every relevant amino acid is accumulated for calculating a representative scoring value. In

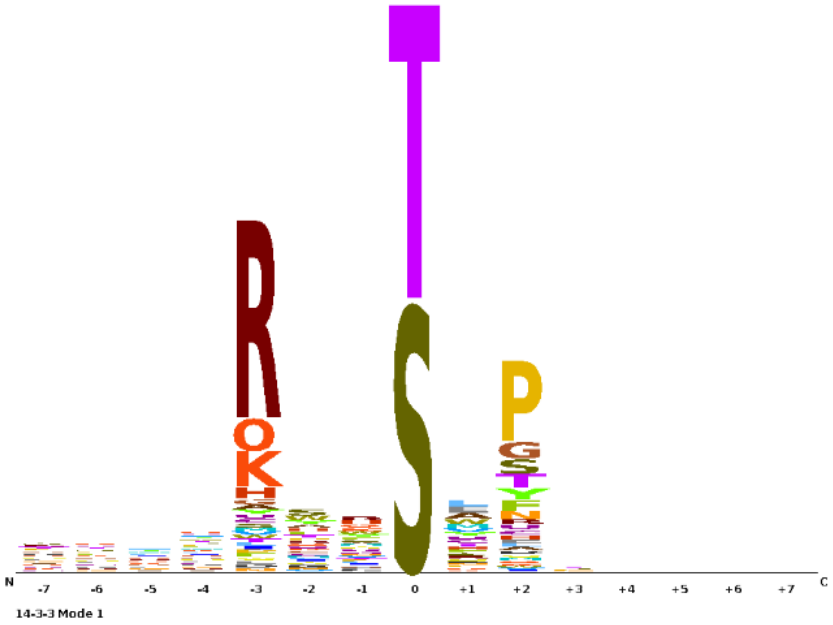


Figure 2.3: This screenshot of *Scansite* 3 depicts the motif visualization of the *14-3-3 protein (zeta/delta)* kinase, which phosphorylates either a Serine or a Threonine (S/T kinase) based on a given pattern. The size of a letter is equivalent to the preference of a residue to appear at a particular position in a motif matrix. As no information about Pyrrolysine is given in this motif, *Scansite* 3 uses the same probability as for Lysine.

this application, a small (good) score means a high likelihood for a peptide to be phosphorylated based on a motif and thus with respect to the motif related kinase. As previously mentioned, *Scansite* calculates its scores based on a numerical representation of motifs, the PSSMs. A *Scansite* score represents the divergence of an amino acid sequence match to an ideal peptide match to a motif. Consequently, a small score is better than a big one. Ehrenberger [9] describes the computation of the relevant scoring values, based on a given PSSM. A residue of a peptide, that matches the zero position of a motif, is called phosphorylation binding site or phosphor acceptor site. However, neighboring residues heavily influence whether the binding site attracts a phosphor molecule or not. In this context it is mostly just referred to as site. *Scansite* scores peptides based on these sites and their surrounding AAs with respect to available (and by the user selected) motifs. In *Scansite* only sites with 7 attached residues on both sides, C-terminal as well as N-terminal, are considered valid peptides. Due to small distances the residues right next to the binding site are supposed to be most influential. As this behavior is already represented by the values of the PSSMs, scoring itself is not required to treat positions differently anymore. The scoring process itself can be described in five steps as follows, whereby any equations were defined by Yaffe et al. [30]:

1. First, a peptide of choice is selected. This usually is an extract of a protein sequence. Original values of a PSSM are selected based on the peptide sequence. In a following step, these selections are transformed using a binary logarithm

as depicted in Equation 2.1. This way particularly small values between zero and two, that are supposed to be disfavored by the motif, are punished more heavily. On the other side, values equal or close to two are treated neutrally and even larger values, which usually represent motif specific affinities, are able to be underlined. Only outstandingly intense values, that are actually relevant for the peptide match scoring process, remain with relatively high values. As a result only the motif specific characteristic values have big positive impact throughout the scoring procedure.

For the first score, a peptide sequence with 7 residues in both directions next to the zero position is processed. Again, this usually is an extract of a protein sequence with S, T or Y in the middle. A scoring window w covers the peptide positions $[-7;-1]$ and $[+1;+7]$, that are relevant for scoring. Additionally, i is an iterator of this window, that represents a line in a PSSM, and aa_i stands for the amino acid value of the peptide on the appropriate window position i. This value is provided by the PSSM, is log_2 transformed and represents each AA of the peptide depending on its position.

$$w_i = \frac{ln(aa_i)}{ln(2)} = log_2(aa_i) \tag{2.1}$$

For the following computational steps, the short peptide *EERsPAK* shall serve as an example. Hereby **s** represents a centrally located Serine as phosphorylation site. To further reduce complexity, the example PSSM, as shown in Table 2.2a, is also limited to the positions $[-3;+3]$, and contains only amino acids of the peptide. Since best possible values are also required in a later calculation step, Table 2.2b lists the columns that contain the best possible motif scores. The scoring function covers all positions except for the zero position. As the motif is a predefined S/T or Y motif, there is no better choice of a residue and thus the binding site is omitted for scoring. Once more, the score is defined as a divergence from the optimal score. As each time *21* would be selected for this particular position, involving it in the scoring procedure would not provide any further information. This can be underlined by comparing the zero positions in the Subtables 2.2a and 2.2b. Consequently, the phosphorylation site can be omitted to save computation time for scoring without losing relevant information. Even though skipping one position seems to be trivial, computation time is actually saved while running a huge number of scoring calculations during a *Scansite* run.

2. After applying the binary logarithm to the scoring relevant positions of the peptide based on the values of Table 2.2a, its values yield for each peptide residue respectively as follows:

   E(0.20)-E(-0.32)-R(0.44)-s-P(4.26)-A(0.68)-K(1.74)

   In the following step a *raw* score is calculated by averaging the logarithmized values as described in Equation 2.2. Hereby n is the number of positions around the phosphorylation site to be scored, which usually is 14 in *Scansite* ($[-7;-1]$ and $[+1;+7]$). For the previous example *EERsPAK* the positions $[-3;-1]$ and $[+1;+3]$ add up to 6. After processing these 6 positions, the raw score for the

Table 2.2: These Tables show generated PSSM values. Instead of using an OPLS matrix, these numbers are calculated by an *in silico* process. For this generation method, the sites of the *PhosphoSitePlus*[16] substrate page of the CDK1 kinase were used. The generation of these values is described in detail in section 3.9. Since these Tables are only of conceptual interest, they are not supposed to correlate with any biological interpretations of real data, although this can still be the case. Whereas Table 2.2a emphasizes the PSSM values of the CDK1 motif with respect to the example peptide *EERsPAK*, Table 2.2b stresses the best possible values of the motif matrix. For the sake of simplicity only the relevant columns are displayed respectively.

(a) Peptide values

	E	R	S	P	A	K
-3	1.15	1.15	1.29	2.20	1.39	0.80
-2	0.80	0.63	1.53	4.18	1.88	0.31
-1	0.98	1.36	1.71	1.71	1.81	0.84
±0	0	0	21	0	0	0
+1	0.14	0.07	0.18	19.22	0.04	0.11
+2	1.15	1.67	1.25	2.16	1.60	1.22
+3	0.70	3.62	1.25	1.71	1.01	3.34

(b) Best possible values

	L	P	S	T	R	K
-3	2.79	2.20	1.29	1.01	1.15	0.80
-2	1.85	4.18	1.53	0.70	0.63	0.31
-1	1.99	1.71	1.71	0.94	1.36	0.84
±0	0	0	21	21	0	0
+1	0.14	19.22	0.18	0.07	0.07	0.11
+2	1.29	2.16	1.25	0.70	1.67	1.22
+3	0.84	1.71	1.25	0.70	3.62	3.34

example peptide *EERsPAK* results in $s_{raw} = 1.17$.

$$s_{raw} = \frac{1}{n} \cdot \left(\sum_{i=-\frac{n}{2}}^{-1} w_i + \sum_{i=1}^{\frac{n}{2}} w_i \right) \tag{2.2}$$

3. Once the *raw* score is available, a reference value is required to get comparable relative scoring values for each peptide of a *Scansite* search. Other than the calculation of the *raw* peptide score, which is processed during the *Scansite* search, a reference is calculated in advance during the database setup and also stored in a motif table of the database. This reference number is defined as the optimal score s_{opt}. During this step a very similar computational mechanism is applied. The are only two differences between the calculation of s_{raw} and s_{opt}. The first one is the selection of values within the same matrix. For the optimal score the best possible values for each position are picked to be transformed by a logarithmic operation as displayed in Equation 2.3.

$$s_{opt} = \frac{1}{n} \cdot \left(\sum_{i=-\frac{n}{2}}^{-1} max(w_i) + \sum_{i=1}^{\frac{n}{2}} max(w_i) \right) \tag{2.3}$$

The second difference is about C- or N-terminal preferences of motifs. In case a sequence motif shows a preference for a terminus, additional calculations are run, which can influence an optimal score. Only if an optimal score with a worse value than a regular amino acid match is found, the optimal score is replaced by the worse one. This worst case optimal score step is required to keep the gap between *raw* and *opt* score at a useful level. A motif, which can be affiliated with terminal sequence matches, is usually only connected to a beginning or to an end of a protein sequence. As a consequence a regular peptide–motif match is not expected and the optimal score has to be adjusted to match a terminal

sequence best. To prevent *Scansite* from assigning actually bad scores to s_{opt}, only the first instance of a C- or N-terminal residue is scored and any further located residues, that would apparently also be terminal positions, are omitted from scoring. As the simple example does not include any terminal affinity, the picked values are transformed by the binary logarithm as follows:

$$L(2.79) - P(4.18) - L(1.99) - s/t - P(19.22) - P(2.16) - R(3.62)$$
$$\downarrow log_2$$
$$L(1.48) - P(2.06) - L(0.99) - s/t - P(4.26) - P(1.11) - R(1.86)$$

By calculating the mean of these transformed values, the optimal score for the generated CDK1 motif is $s_{opt} = 1.96$. With s_{raw} and s_{opt} as input, the actual *Scansite* score can be calculated as described in the following point.

4. Eventually, a final score s_{final} is calculated by setting the previously ascertained scores in relation as shown in Equation 2.4. For the CDK1 example, *Scansite* would display a score of 0.40. Based on the final score formula, low scores represent good results, whereas high numbers are equal to worse ones.

$$s_{final} = \frac{s_{opt} - s_{raw}}{s_{opt}} \tag{2.4}$$

A better score for a lower value can be interpreted as less difference to the perfect peptide match to the motif, whereas bigger scores represent a less fitting amino acid sequence with respect to the motif. The final score can be assigned three different values or ranges. The following list provides more detailed information about possible scores ordered by theoretical frequencies:

(a) Positive: A regular *Scansite* score is a value > 0 in the range [0;5]. The s_{opt} score is usually greater than the s_{raw} score. Hence a positive final score is most common and can be defined as regular score for peptide matches.

(b) ± 0: If s_{opt} is equal to s_{raw}, a perfect match is found. Although this is very unlikely to happen, it is still possible. Of course a perfect match would only be an option, if the protein sequence extract is the same as the ideal AA sequence of the motif.

(c) Negative: The last and least probable result is a negative score. For regular peptide–motif matches such a score is not possible at all. However, if a motif with terminal preferences matches a peptide with an ideal non-terminal sequence, chances are there, that s_{raw} is actually greater than s_{opt}. The result would be a negative score. A negative value can also be considered as a perfect match.

Since extremely unfortunate raw scores can result in big negative numbers, there is – in theory – no maximum value for a score. *Scansite* however, will not show numbers above a certain threshold, since they are discarded right after

calculating [9]. The threshold depends on a parameter called *STRINGENCY* which can be set by the user. Possible values are *HIGH*, *MEDIUM*, *LOW* and *MINIMUM*. The lower the *STRINGENCY* value is set, the more results are accepted for the *Scansite* result list. The connection between specific *STRINGENCY* values and the results of a *Scansite* search can be described as listed below.

- *HIGH*: Only top 0.2% of scored sites are included in the returned results
- *MEDIUM*: Only top 1% of scored sites are included in the returned results
- *LOW*: Only the 5% of scored sites are included in the returned results
- *MINIMUM*: Top 15% of scored sites are included in the returned results
- General cutoff: Scores worse than 5 are discarded immediately

5. Unfortunately, a score can only be used to compare sites with respect to one specific motif. To be able to compare site scores related to various motifs to each other, one more step is required. For this purpose, another scoring value, the *percentile*, is calculated. As this final value is a comparative value, calculations have to include essentially all possible references. Scores for any site of the entire proteome, that are covered by the databases, are precalculated. Since a huge number of scores is given, it can be seen as part of a distribution, ideally a normal distribution. Consequently, the *percentile* is a value, that is based on this very population of potential scores. As one can not simply expect a normal distribution to be given, a so called modified z-score is calculated instead of a p-value. The calculation of this number, that sets a site in relation with the entire proteome, is displayed in Equation 2.5.

$$z = \frac{s_{final} - \tilde{x}_{RP}}{c \cdot \mathrm{MAD}_{RP}} \tag{2.5}$$

Hereby, $\tilde{x}_{RP}$ represents the median of the reference proteome and MAD_{RP} is the median absolute deviation (MAD), also with respect to the reference proteome [9]. The variable c is a constant value (1.4826), which is defined as normalization factor [27]. To be more specific, Rousseeuw and Croux [27] make clear, that this value is supposed to ensure consistency for the estimator of the parameter of interest.

Even though methods have different advantages and downsides, especially calculations based on the median and the MAD are particularly reliable, since these values show a very satisfying behavior with respect to robustness. Hampel et al. [14] point out a number of arguments, that strongly underline this attribute with properties such as a fairly low breakdown point. To compare the results more easily to the reference proteome, *Scansite* generates a histogram (Figure 2.4). The diagram is available in every result table entry of all major *Scansite* features. To get a good impression of most relevant values, the final score s_{final} (displayed as *"Score"*), the median of the reference proteome (displayed as *"Median"*), the MAD (displayed as *"Median Abs Dev"*), as well as curves of absolute frequency and percentile are depicted in the diagram (Figure 2.4). By evaluating the absolute frequency curve using the *Score* as its

x-coordinate, the z-value, which calculation is shown above in Equation 2.5, is defined as its y-coordinate (almost zero in the example below).

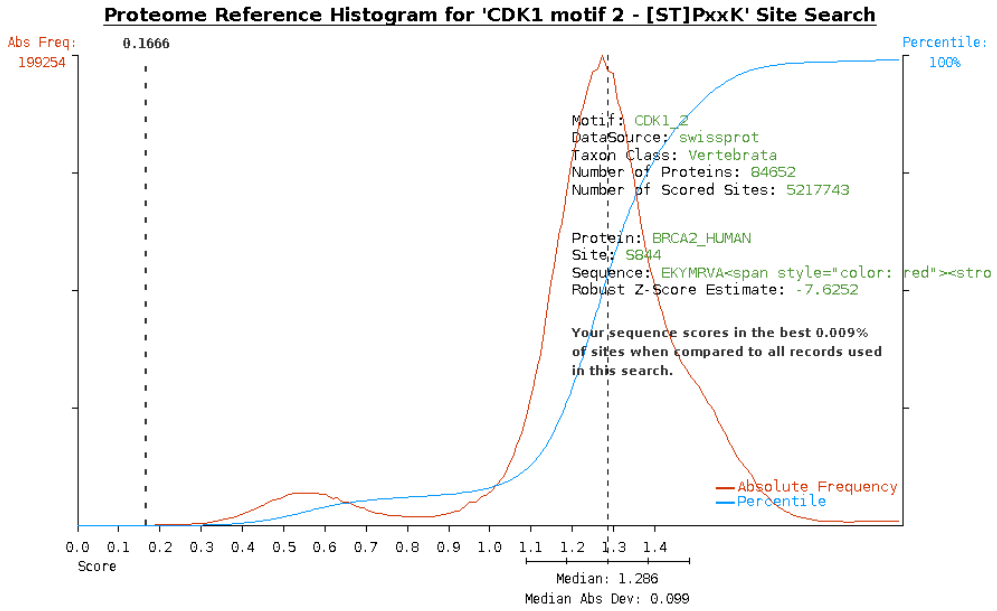


Figure 2.4: This *Scansite* histogram is generated by matching the site location *S884* in the protein *BRCA2_HUMAN* (matched site sequence *EKYMRVAsPSRKVQF*) to the in *Scansite* available motif *CDK1 motif 2 - [ST]PxxK*. Next to some database related values, the score is displayed at 0.167 which is – being close to zero – a very good score. The value of a *z-score* can be determined by the value on the x-axis where score and the percentile curve intersect. Since the value in this case (0.009%) is extremely small, the matched site is highly unlikely to be a random result. Thus this very peptide match is excellent in *Scansite*.

Up to *Scansite* 3 it is possible to score sites with respect to 70 available sequence motifs. Since calculating site scores based on the entire proteome would be impossible in real time, many calculations, such as previously mentioned optimal scores of motifs or the rendering of histograms, are done in advance. Usually, motifs are added to the database right after mirroring publicly available databases, that are described in detail below. During this process of inserting motifs, any additionally required data is being generated.

2.3.1 Databases

Although *Scansite* is a standalone web application, it heavily depends on data from various sources and even different types of information. Hence a MySQL database, which stores any required information in data tables, builds the foundation for the application. To retrieve results quickly, *Scansite* and the MySQL database interact based on prepared statements written in structured query language (SQL).

As previously mentioned, a *Scansite* database is set up by mirroring publicly available databases. In general, three different types of databases can be distinguished: First, and most important, there are protein databases, that include information

about proteins like identifiers, accession codes, sequences and more. All this information is highly relevant, since newly developed drugs have to target certain proteins or other specific molecular groups. Thus especially phosphorylation binding regions, which control particular cellular mechanisms, form a point of interest. The second category is about orthology databases. These are relevant to find evolutionary conserved sites between proteins, and consequently relations to and between motifs. Finally, there is also a single database, that provides information about the localization of proteins within a cell, which can be important for figuring out treatments. Depending on the location within a cell, transport mechanisms of potential drugs have to be adjusted.

Protein databases can be seen as the core database type, as they are absolutely essential. Basically, every single feature of *Scansite* relies on this central group of databases. Apart from different sources, there is also a variety of organisms, data is available for. Data sources provide information for species such as human, mouse and yeast. To be more specific, the mirrored protein databases include the following ones:

- UniProtKB/Swiss-Prot
- TrEMBL
- Ensembl Human
- Ensembl Mouse
- NCBI Protein
- *Saccharomyces* Genome Database (SGD)

Besides, these core databases there are also ortholog databases in use, namely“SwissProt Orthology” and “NCBI HomoloGene”. These two data sources are also required in *Scansite* features and include evolutionary conserved phosphorylation sites of various species. As a consequence, relations between different organisms can be determined, which in turn leads to a further understanding of biological mechanisms. At this point, there is one specific function called “Scan Orthologs” that uses the databases directly. This ortholog scan, which will be explained in section 2.5.4, can also be applied to results of other *Scansite* searches.

The third and last category of data used by *Scansite* contains information about sub-cellular locations of certain molecules, particularly kinases and their substrates. The data of the “Protein Subcellular Localization Prediction System” *Loctree 3* is provided by Goldberg et al. [13]. This data source is used in major *Scansite* features. Based on the results, it is possible to tell whether a matched protein and a kinase of choice are supposed to be localized in the same sub-cellular position. If they are not, it is very unlikely for the kinase to catalyze a phosphorylation process.

To retrieve information from all those databases, relevant information about the sources is configured in *eXtensible Markup Language* (XML). Additionally, an XML schema enforces the correct structure of the configuration file. Apart from the database mirroring, there is more information, that is required to be stored in the *MySQL* database.

2.3.2 Motif Data

The core of *Scansite* is composed of motif PSSMs, which are also listed in an XML configuration file. However, the XML document only references plain text files, that contain the actual preference values for each amino acid with respect to every position in the motif. Apart from those references, the XML configuration also includes further information about the motif, which can be split into 7 points as described below. Whereas only describing certain elements is usually very abstract, an example is often able to create a connection to the content. Thus a shortened *motifs.xml* file as depicted right below describes the general structure of effectively any motif configuration. The XML extract shows the overall structure starting with meta data, and at least one content related *Scansite* insertion entry. Hereby, its content has been shortened to a single sample motif. The comment *further motif entries* should point out, that usually multiple motifs are covered within a single *motifs.xml* file. The remaining entry lists all relevant elements, that are required for inserting the motif *14-3-3 Mode 1* into the database.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE motifInserter SYSTEM "Motifs.dtd">
3 <motifInserter>
4   <motifs>
5     <motif>
6       <shortName>1433\_m1</shortName>
7       <displayName>14-3-3 Mode 1</displayName>
8       <groupShortName>pST\_bind</groupShortName>
9       <groupDisplayName>Phosphoserine/threonine binding group</groupDisplayName>
10      <motifClass>MAMMALIAN</motifClass>
11      <isPublic>1</isPublic>
12      <identifiers>
13        <identifier>
14          <name>YWHAZ</name>
15          <typeId>4</typeId>
16        </identifier>
17        <identifier>
18          <name>1433Z\_HUMAN</name>
19          <typeId>6</typeId>
20        </identifier>
21      </identifiers>
22    </motif>
23    <!-- further motif entries -->
24  </motifs>
25 </motifInserter>
```

Lines 1 and 2 are independent of the information part of the file. These lines just define language specific settings and a document type definition (DTD), that is referenced. By using a DTD, correct XML syntax and content can be enforced. All the following lines 3-25 provide the actual informative content of the file. Any elements are described in more detail right below.

1. **shortName:** This element represents the short name of the kinase motif, which is used by *Scansite* to find the actual motif file. Moreover, the short name is used for most internal processes, particularly during searches.

2. **displayName:** A display name is used in the web interface of *Scansite*. Especially selection boxes throughout the application, which display the names, are more informative, since clearly defined and commonly known names are used.

3. **groupShortName:** Similar to motifs, also motif groups have short names. As previously mentioned, short names are primarily used for internal processes of *Scansite* such as database queries and other mechanisms, that are relevant for scans.

4. **groupDisplayName:** Because some *Scansite* searches offer a selection of particular motifs or motif groups, a displayed group name is required. This display name is visible in all motif group selection boxes, that are available in search forms. Although any motif name should be able to speak for itself, the "Databases and Motifs" section lists all motif groups together with the motifs they include. There are 14 motif groups, that are currently available. Each and every group is defined by certain attributes, which make the group unique. Although the groups itself can be clearly distinguished, assigning motifs can still be challenging due to multiple suitable features. At this point, only the groups, that are listed below, are available for any *Scansite* user. Even though the number of motif groups is currently limited, new ones can easily be introduced to the system. Once a new group appears in a motif configuration file, it is automatically recognized by the system and inserted into the database. Currently motif groups are available as listed below.
   - Phosphoserine/threonine binding group
   - Tyrosine kinase group
   - Src homology 2 group
   - Src homology 3 group
   - Basophilic serine/threonine kinase group
   - DNA damage kinase group
   - Acidophilic serine/threonine kinase group
   - Proline-dependent serine/threonine kinase group
   - Kinase binding site group
   - Lipid binding group
   - PDZ domain binding group
   - Phosphotyrosine binding group
   - Hydrophobic-directed serine/threonine kinase
   - Other serine/threonine kinase
5. **motifClass:** The motif class defines whether the motif should be classified as *"MAMMALIAN"* or as a *"YEAST"* motif. At this point there are no other existing motif classes, as the current motif classes are sufficient for any motifs provided by *Scansite*. However, it is possible, to easily add further classes to the system, if it is required in future.

6. **isPublic:** In *Scansite* 3 preparations for publicly available motifs and private only ones were made. These preparations include a flag in the XML configuration of the motifs. Every motif has either 1 (publicly available) or 0 (private access only) as value for the *isPublic* element. The term *preparations* is used, because the availability of private motifs based on a user login was introduced during the development of *Scansite* 4. This means that – starting with *Scansite* 4 – a user, who just uses the application, is able to use all features together with all publicly available motifs. If a person has a *Scansite* account, it is also possible to log in to execute searches with different privileges. Once a user is logged in, also the private motifs are available, which offers more choices on the input side, and more or different results on the output side. A *Scansite* account is not available for any user. As private motifs are in test stage or part of active research, access is restricted and an account registration is not available. New accounts can only be added by contacting a system administrator.

7. **identifier:** Each motif configuration usually covers two different identifiers. The motifs XML file currently includes only two types as specified below. The Scansite system recognizes those types, and the Scansite database provides any necessary information about the identifiers. *Scansite* relies on these identifiers to be able to map a correct identifier to a protein in the right database table. Despite identifier mapping is an issue in bioinformatics in general, due to many different sources and their standards and identifiers respectively, *Scansite* deals with multiple formats simultaneously by recognizing and using more than a single identifier. By specifying types for each and every identifier in the system, *Scansite* accomplishes correct mapping of proteins and databases. This also includes the kinases, that are related to the motifs, as the two identifiers in a motif definition are those related to its kinase.

   - **Type 4** represents a so called HGNC gene symbol. HGNC is the abbreviation for *HUGO Gene Nomenclature Committee*, which ensures a standardized nomenclature in human genetics [23]. The HGNC gene symbols are used in *Scansite* together with motif definition files to identify and reference proteins of certain data sources correctly.

   - **Type 6** defines a *UniProt Entry Name*, which is – similar to the HGNC gene symbol – a common and representative ID for proteins. Usually the *UniProt* entry names contain an acronym representing the protein itself combined with its related species, e.g. *BRCA2_HUMAN* [24]. As *UniProt* names are very common throughout different types of databases, some *Scansite* features are limited to this type of identifier.

   Although there are more identifier types (1: Ensembl Protein ID, 2: Ensembl Gene ID, 3: GenBank Protein Accession, 5: SGD Systematic Name), type 4 and 6 are most common. Thus these types are currently the only ones, that are related to motifs or more specifically to the kinases on which the motifs depend. By storing multiple identifiers in its database, *Scansite* is able to run inquiries based on different databases, even though it is only possible to deal with a single identifier type at a time.

2.3.3 Evidence Data

Another type of third party input is called *evidence data*. Similar to handling motif data, evidence entries are handled by a central file. Instead of referencing plain text files, the actual evidence data is available on the web. Essentially *Scansite* recognizes any site evidence based on given IDs in the database and links to actual evidence web pages as part of its results. Any content of the evidence file is split into a basic web address and specific IDs, that are able to link to designated references. This structure of the evidence file allows to keep entries from three different sources called:

- **PhosphoSitePlus** [16]: http://www.phosphosite.org/homeAction.action
- **Phospho ELM** [8]: http://phospho.elm.eu.org/
- **Phosida** [12]

To answer the question, what these previously mapped phosphorylation sites are good for, a basic concept on *Scansite* needs to be explained. As the application comes up with potentially phosphorylated sites based on motifs and protein sequences, it might contain multiple sites, that are already known, anyway. Based on this concept, it could be possible, that the software comes up with random or simply incorrect matches, that are absolutely unrelated to the motif and/or the protein of the scan. *Scansite* however, uses these known phosphorylation sites to underline its own results. Next to any matches of a search, *Scansite* also lists previously mapped sites for each result entry, if it exists. With multiple references of a single search, the program emphasizes the likelihood of its results to be correct and thus its connection to *in vitro* and *in vivo* research. As a consequence, by using the evidence data, *Scansite* matches are underlined to have real value instead of coming up with random matches.

2.4 Workflow

As *Scansite* uses *Google Web Toolkit (GWT)*, remote procedure calls (RPCs) have to be handled each time, a request is sent to the server. The application covers several features together with its complexity. Hence a defined process in the architecture of *Scansite* is essential for structured processing of RPCs. Birrell and Nelson [4] explain, that a procedure call is a well-established mechanism regarding transfer of data like parameters or certain requests. Although this behavior is very handy, a remote procedure call includes asynchronous characteristics, which makes things more complex. Once a request has been sent, the current thread[1] has finished its job and cancels the connection to the server to return to user specific tasks or to terminate. At the same time another part of the application is responsible for receiving the response, once it is sent from the server to the web client. Such a response usually contains result data of a search. *Scansite* has run searches on the server side, where it is able to access the database and to work through the scoring process.

[1] A computer can execute multiple tasks simultaneously. If some of those parallel tasks belong to the same application, each of these processes run in its own thread, its own time line for executing specific commands, while other things run at the same time.

All this network related interaction requires a properly engineered system, that is able to deal with client-server messages without causing big latencies or a computational burden to the system. At the same time implementation efforts should be kept at a minimum for newly introduced features, that are currently available or potentially in future. With all those points in mind, *Scansite* is split into three mostly independent parts. These sections of the software are defined as (i) server side part, (ii) client side part and (iii) a shared section. Whereas the server side application is running in *Java* on the physical *Scansite* server, the client side part is translated into *JavaScript* and is run in a web browser. The shared fragment primarily contains serializable containers, that are able to send data from the client to the server and the other way round. This superficial view on client–server interaction does not tell anything about the relatively complex processes on the technical side. Hence a more detailed view is necessary, that is able to explain the structure of *Scansite* on both, server and client side.

2.4.1 Regular Server–Client Interaction

A regular interaction is used to perform simple functions such as loading standard pages of *Scansite*. For instance, such an interaction is required for loading the "News" page or any other rather trivial page, that has simple call routines to request data from the server. How such a call mechanism works, and how information is eventually displayed in a web browser, is explained separately for client and server part, as these have to fulfill different tasks.

Client Side

On the client side information is essentially requested and displayed. These assignments are split into different entities of the software. Each of these entities will be referred to as *class* and related code for specific purposes will be put together in such a unit. Based on this definition, the client side in this scenario has 7 different classes, that are responsible for receiving and displaying data. Whereas some classes are managing traffic and data handling, others are in charge of filling containers, displaying data or assisting in any of those assignments. Following, the purpose of every class will be elucidated.

- Presenter: The presenter class can be described as the controller class, which manages all data handling and the RPCs on the client side. This unit is the core of each interaction cycle. Due to its controlling job a presenter processes the user input (e.g. loading request for another page of *Scansite* via mouse click), requests required data, sends necessary parameters and also receives any results. Once the presenter class possesses all needed information, the data gets distributed to other classes, that are responsible for displaying information properly.
- View: The view is the only collection of code, that is not called *class* but *interface* instead. An interface does not necessarily provide defined and structured functionality. Major tasks of this unit include defining the structure of a class and only provides basic functionality. Based on the definition of an interface, a class can be implemented on top of it. This class inherits all definitions and any (basic) functions of the interface.

- View Impl: The *View Impl.* or *View Implementation* is such a class, that uses the view interface as its base. Consequently an implementation class makes use of any predefined variables and functions, that are already available. On top of that this class also provides further functionality, that handles most of the work. The view implementation also servers as a container, that comprises and prepares elements, that need to be filled with the result data.
- Widget: Those elements are primarily widgets of *GWT*, which are in charge of displaying information. Widgets can be tables or defined graphical structures, that are able to interact with the user like input forms for search parameters. These elements can also be set up in a hierarchy. Small widgets can be defined and combined in a bigger common one.
- Ui Xml: This component is mostly defined in XML and provides the basic HTML structure, that is responsible for the visual output in the web browser. Moreover the user interface XML also defines static components like headings or other fixed text elements.
- Action: An action is triggered by the presenter class. This unit contains all relevant information that is required for server side jobs. Consequently an action is in charge of transmitting data to the server, which fires the server side process routine to provide results respectively.
- Result: The result class serves also as a container for specific information. In contrast to the action component, a result – as the name says – contains any results, that are provided from the server. Additionally, this class is in charge of transmitting its content to the web browser on the client side, where a presenter distributes any information to target elements accordingly.

Server Side

Whereas the client side has many classes, since the presentation of the data is also handled there, the server side can fulfill its job with three types of classes for simple inquiries. On the server side a request is received, processed by doing calculations and accessing the database, and eventually a response in form of usually search results or other needed data is returned. As these processes run independent of the client, calculating results and preparing data is not connected to the power of the computer of the user at all. This allows running *Scansite* even on mobile devices, if necessary. Server side classes are defined as follows:

- Handler: A handler class is primarily responsible for handling traffic with the web application on the client side. It – as the name says – handles incoming requests and delegates tasks to data access objects (DAOs). Once the computational part on the server side is finished, a handler sends the requested information to the web client.
- Dao is the term for a layer, that is responsible for accessing data in general. It provides a certain set of functionality whereby classes such as the handler which use a DAO do not have to care about the data source. It could be a plain text file, a XML document or a database or basically any other source, that is able to provide required data. In *Scansite* all DAOs are engineered to access a *MySQL* database to retrieve any relevant information. Thus the DAOs make use of other predefined command classes.

- Command: A command class contains information about the tables and columns, that have to be accessed to retrieve specific data. Each command also provides a in *Scansite* unique SQL query called *prepared statement*, which can be used for accessing the database quickly. A prepared statement is a SQL query, that is allocated in advance instead of just in time, which makes its execution and thus the database transaction faster.

Meta Classes

Next to the classes, that are described above, the interplay between all those client and server side actions and in between has to be configured. Meta classes define, which set of client entities is in charge of dealing with a click or typing event, a user fires and which ones on the server side have to deal with inquiries of the web application. In general, there is one class for each side: The presenter factory on the client side and the actions module on the server. Task assignments of both meta classes can be defined as described below.

- Presenter Factory: The presenter factory is the core controlling mechanism on the user side. This class defines a navigation bar and its events and destination pages. It is also responsible for setting a presenter and a view correctly. The other five client side classes depend on those settings too, although they do not have to be set explicitly, as they are woven into view and presenter respectively. In the presenter factory regular and result pages are also distinguished and the behavior while waiting for results is managed as well in this component. Next to handling visual effects like stressing the currently active element in the navigation bar, also administrative and usual page access is differentiated.
- Actions Module: The actions module class does not have to deal with visual elements. As a consequence, it only has to define which action needs to be bound to a particular handler.

The interaction of the three types of classes is visualized in Figure 2.5, which depicts the processes of a single trivial user action such as loading another page. This visualization specifically shows, how classes interact and in what order the call routine is executed.

2.4.2 More Sophisticated Server–Client Interaction

As the above described interplay model is rather trivial, it is not sufficient to deal with more complex tasks such as particular search requests. It is possible to load any page of the navigation bar using the above mentioned simple routine. However, time consuming and iterative jobs require a more complex system and thus additional classes, which are together with the previous functionality able to deal with a higher level of complexity. There is one additional class on each side, an *Event* class for the client side and a *Feature* class for the server. Their purpose as well as additional functionality can be explained as follows.

- Event: An event is required, if based on received data, another server request needs to be triggered. The presenter, which receives the results, triggers an event, which again launches another client–server routine. The event delegates its task to the presenter factory, which instantiates another presenter entity,

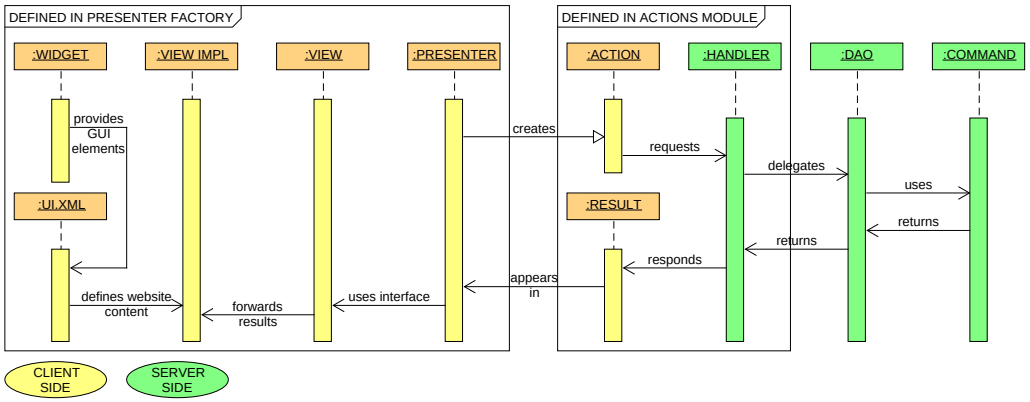


Figure 2.5: This sequence diagram describes a rather trivial *Scansite* client–server interaction. Based on user input, a presenter factory selects the correct presenter and view. The presenter instantiates an action, which receives any request specific parameters and is eventually sent to the server. As soon as the server receives the action, the actions module selects the correct handler, which anon processes the request by launching data access objects (DAOs). Those DAOs use commands to access the database. All retrieved information is returned to the handler, where data is processed and finally sent back as a result class instance to the web client. The web application receives results through its presenter class, where data is distributed to its view and other elements such as widgets, that are part of the view. Once all data is set, the presenter factory displays the requested (result) page.

that again interacts with the server. All results are returned through the event back to the original presenter instance. At this point results are distributed to designated destinations and the job of the event is completed.

- FEATURE: A feature is introduced between handler and the DAOs, to create another abstraction layer, which reduces complexity. Especially for *Scansite* searches, that consume a lot of server resources, multiple different types of data are required and more complex methods have to be run. This additional complexity is defined in a separate feature class, to keep it separated from simple processing, network interaction and database access.

The two additional classes and their interplay with all previously described ones of the trivial client–server interaction are shown in Figure 2.6. Hereby the diagram is still simplified. A single feature usually requires multiple database DAOs to perform a search and to receive all information for calculating scores.

2.5 Provided *Scansite* Features

Scansite is a software application, that offers a variety of different features. Some sub pages simply provide information such as news or data specific elements like an overview of particular structures, e.g. motif logos. Other pages offer input forms, which represent the actual *Scansite* program. Those pages all require the more complex client–server interaction, as more challenging computational operations are required to be executed. The basic functionality, which is partially also explained by [10], is defined in the following sections.

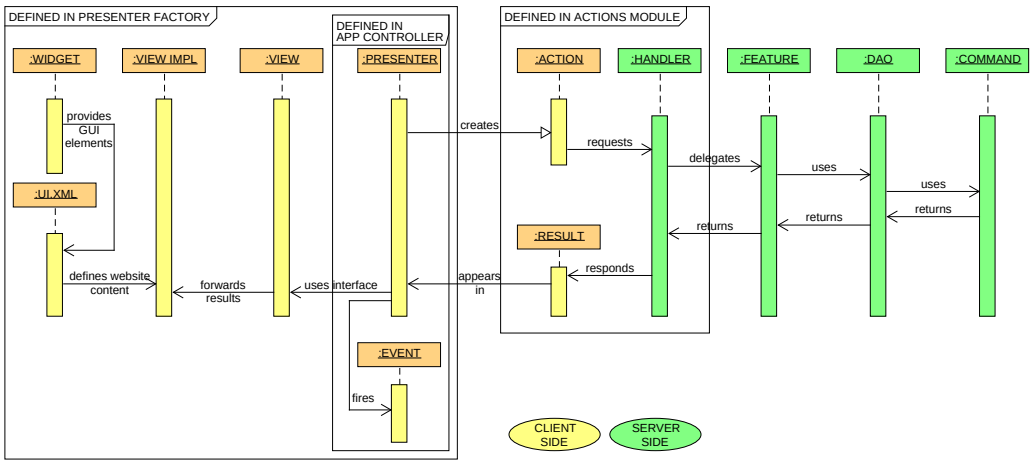


Figure 2.6: A more complex client–server interaction is depicted in this second sequence diagram. Compared to the trivial version in Figure 2.5, additional classes are required. An event is triggered by a presenter, if another interaction iteration required. In this case an event fires a routine, that sends one more requests to the server and receives results respectively. On the server side a higher complexity is also essential to complete required jobs. *Scansite* searches often require different types of data, which need to be handled by multiple DAOs within a single feature. Also processing information and providing results for its handler is done by a feature.

2.5.1 Scan Protein for Motifs

Feature and Input Options

The first of the two *Scansite* core features is the protein scan. With the "Scan Protein for Motifs" option, the software is able to match a specific protein to one or multiple particular motifs or motif groups. By doing so, *Scansite* finds potential phosphorylation sites and consequently potential substrates. Based on this information conclusions on further regulation mechanisms can be drawn. Especially by combining searches for proteins of a healthy person and one with a disease, causes for an ailment can be discovered.

This feature offers multiple search options, which are able to adjust a scan according to the preferences of a user. Depending on whether a publicly known protein or a user defined one is preferred, *Scansite* offers to choose either one or the other option via radio buttons on the web form. Depending on which option is selected, it is either possible to select a public database, that is supposed to contain the protein of choice and enter the identifier below, or a name for the protein together with its sequence can be inserted into text fields. After the selection of a protein, one or multiple motifs have to be selected. There are the following options to do so.

- **Select by motif class:** The easiest option, which is also the default one, is choosing the *MOTIF SPACE "All motifs"*. Within this motif space all motifs from a particular motif class are automatically selected to be matched to the protein of choice. At this point only the 70 Mammalian motifs or the 54 for Yeast can be selected.

- **Select one or multiple specific motifs:** By choosing the second *MOTIF SPACE "Selected motifs and groups"*, a more detailed choice can be made. It is able to select a single motif by clicking on it in the list. If the *Ctrl* key is pressed, additional selections via mouse click are possible.
- **Select one or multiple specific motif groups:** Similar to the selection of particular motifs, also motif groups can be selected. In the second box, that lists motif groups, it is again possible to select a motif group per mouse click and also a multi-selection is possible by pressing the *Ctrl* key while clicking on additional choices.
- **Select specific motifs and motif groups:** Since the selection mechanism is identical for both, motifs and motif groups, it can be combined. Such a combination with its related highlighting in the boxes can be seen in an arbitrary example, that is displayed in Figure 2.7.

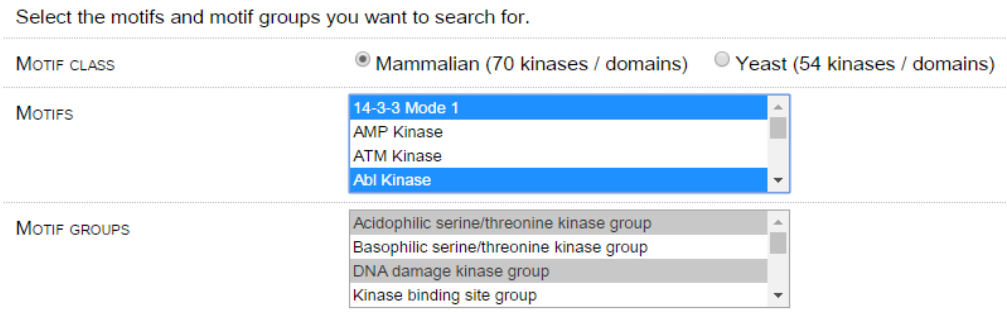


Figure 2.7: An arbitrary example for a combined multiple selection of both, motifs and motif groups is depicted. This mechanism allows to run very specific searches, in case only certain motif categories are of interest.

- **Upload a user defined motif:** Last but not least it is also possible to upload a correctly formatted file, that includes a user defined motif. Next to the file only a name for the motif is required.

Another input parameter for the search is the *STRINGENCY*, which was described in section 2.3, point 4. Depending on which value is selected, the sensitivity and consequently the quality of the results differ. It is highly recommended, not to change the default setting *High*, unless no or no useful results can be retrieved with the default stringency.

The next parameter *SHOW PREDICTED DOMAINS* is an option, that can be enabled or disabled by changing the state of its connected check box. If the parameter is enabled, a third party software called InterProScan is executed additionally to the regular search. This tool is able to predict protein domains and protein families by using a *FASTA* formatted protein sequence as input [19].

For the two remaining *Scansite* protein scan parameters, the *reference proteome* and the *localization data source*, it is recommended to keep the default values. At the moment the localization data source can not be changed, as there is no other available data source for finding subcellular locations anyway. The only alternative for the reference proteome is choosing *Baker's yeast* over *Vertebrata*.

Result Page

After finishing a protein scan, the appropriate result page is displayed in the web browser. Hereby its structure is defined in five independent blocks.

1. **Protein properties:** Information about the protein of choice is displayed. This includes the identifier, further description, accession codes, keywords, molecular weight, isoelectric point and subcellular localization.

2. **Scan properties:** This segment of the page provides an overview of all the search settings, that were described above.

3. **Protein plot:** The protein plot is split into two separate ones. One plot shows all phosphorylation sites along the positions in the protein sequence. If the protein domains are requested, those are also displayed alongside this graph. Right below the first one, a second plot displays the surface accessibility based on the amino acid sequence of the protein.

4. **Predicted motif sites:** This part of the result page forms a table, that contains the best *Scansite* results, which are ordered by motif-peptide match based scores. All motif matches for the protein of choice together with characteristic values are listed within this important table. The protein scan result table displays its data in eleven columns as described below.
   - **Score:** The first column shows the *Scansite* score, that gives a first good impression. In the protein scan however, multiple motifs are matched to a single protein. As a score depends on a single motif, it is only possible to compare the quality of all matches in the result table, that are related to the same motif.
   - **Percentile:** To compare all results to each other, the percentile value can be used. As described in section 2.3, point 5, this values take the entire proteome into account, which makes it possible to compare the *Scansite* score together with its percentile to other entries in the result table.
   - **Motif:** This column lists the display name of the motif followed by its short name in brackets.
   - **Motifgroup:** As for the motifs, the motifs group column shows the full display name and the short name of a motif group in brackets.
   - **Site:** The site column entry contains two elements, the one letter code for the amino acid, that is potentially phosphorylated, based on the activity of the kinase related to the motif, and the position of the AA within the protein sequence.
   - **Sequence:** The sequence, that is listed in this column is the 15 position window (the peptide), that is scored with the motif PSSM. Since *Scansite* 4, the central position of the sequence is stressed. The font of the single lower case letter of the sequence is bold and the color is set to red.
   - The **Surface Accessibility:** is a value between 0 and 17.6 whereas a higher value means, that it is more likely, that a specific region of the protein is on the surface and thus surface accessible.

- **Gene Info:** In this column, information about the genes of interest is provided via hyperlinks to other websites. Examples for linked pages are *GeneCards* which provides gene-centric information [25] or *UniProtKB*, which is one of the biggest databases about protein sequences and associated detailed annotation [5].
- **Previously Mapped Site:** Not every result has an entry for this column, as this information is limited to the previously in section 2.3.3 described evidence data.
- **Evolutionary conservation:** This column offers the opportunity to run an ortholog scan for a particular result. The ortholog scan is another *Scansite* feature, which is described further below in section 2.5.4.
- **Colocalization:** The last column of the table provides information about subcellular localizations. The displayed location is related to the kinase, that is connected to the motif. To additionally provide information whether the location is very likely or not, the text is displayed in a different color. This color depends on the location of the phosphorylation substrates. If the substrates are supposed to be located at the same place as the kinase, the color of choice is green. In any other case, the text is displayed in red. Hereby all information is provided by *Loctree 3*.

5. **Additional analyses:** The final part of the result page offers to download the search results as tab-separated files, to do further analysis independent of *Scansite*. The last section is also able to score the phosphorylation sites in another way. By clicking on the link provided, a new page opens and *DisPhos*, another web-based software for phosphorylation sites based on proteins, analyses the results based on the previous search [18].

2.5.2 Search a Sequence Database for Motifs

The second core feature of *Scansite* is the option called “Search a Sequence Database for Motifs”, which will be also referred to as database search.

Feature and Input Options

It works almost like the protein scan. However, the major difference is that the one to many relation is flipped. This means, that only a single motif is selected, whereas an entire protein database provides protein information to find matches and to generate scored results. As multiple proteins are now able to be processed with respect to a single motif, default *Scansite* scores are way more meaningful. Due to changed proportions of the search, input parameters and results need to be adjusted to new needs accordingly. Especially running a search for an entire protein database can be very expensive regarding runtime and use of hardware resources. Thus multiple ways of restricting a scan are available by use of appropriate parameters as listed below.

- **Choose search method:** Even though only a single motif can be picked for running the search, *Scansite* offers three different ways to select this particular motif.

1. Database motif: The first option is the default setting, which offers to select a Mammalian or a Yeast motif out of a drop down menu.
2. User defined motif: A second way to choose offers more flexibility. If a user meets the requirements of the *Scansite* motif structure, it is possible to upload a user defined motif. These non-database motifs are only used for a single search and discarded right after.
3. Quick motif: For more simple tests or for searches, that require a less complex and yet specific amino acid pattern, a quick motif can be inserted. Up to 15 text boxes can be filled with one letter codes for amino acids or with wild cards, that are described right below the text boxes. These values are treated as values of preference and calculation of scores only considers peptide matches to the sequence of the quick motif.

- **Protein data source:** Since only a single database is searched, it is highly important to select the data source of choice. Hereby *UniProtKB* is selected per default. Additionally, other databases as introduced in section 2.3.1 are eligible.
- **Restriction parameters:** To reduce runtime and use of resources, multiple restriction parameters are available in *Scansite*. The organism class selection is always enabled and uses *Mammals* as default value. There are 7 alternative values as well as the option to pick all organism classes. However, one should refrain from picking all classes due to increased computation time. Next to this first parameter, it is optionally possible to enter two further input filter values, molecular weights and isoelectric points of proteins, to user defined ranges. Proteins, that are not included in the windows, are discarded before the scoring process starts. Whereas it is fine to use the default value for the number of phosphorylated sites, entering a species restriction (e.g. *homo sapiens*) is highly recommended. This very parameter makes the biggest difference in runtime. This behavior is absolutely reasonable, if one considers, that any protein related to another species (e.g. mouse) is omitted for the scoring process. Further restrictions like specific keywords or a particular sequence pattern can also help to drastically reduce computation time.
- **Output list size:** As a database search tries to match and score each and every single protein, usually not all results are able to be displayed on the result table of the output page. Hence the user can choose how many results to list. Available options are 50 (default), 100, 200, 500, 1000 and 2000.

Result Page

Compared to a protein scan, a database search has a very similar structure regarding result pages. This feature limits its output to four sections, that display search settings and results.

1. **Scan properties:** The purpose of this block is identical with the one in the protein scan. Selected search parameters are displayed in an overview. Also parameters, that have not been set previously are displayed together with information such as *not set*.

2. **Scan result properties:** This unit displays values with respect to the database and the search settings in particular. Figure 2.8 displays an example search using the *14-3-3 Mode 1* motif and the species regular expression *"homo sapiens"* (without quotes). This image shows how many proteins are in the protein database, which in this case is the *UniProtKB*. More importantly it shows, that only about 3.6% of the entire database needs to be processed just by restricting the species. During the processing only a limited amount of proteins actually shows potential phosphorylation sites, that are relevant for calculating a score in *Scansite*. Also a median of all scores and its related MAD are displayed to give a first impression of the results before going into detail.

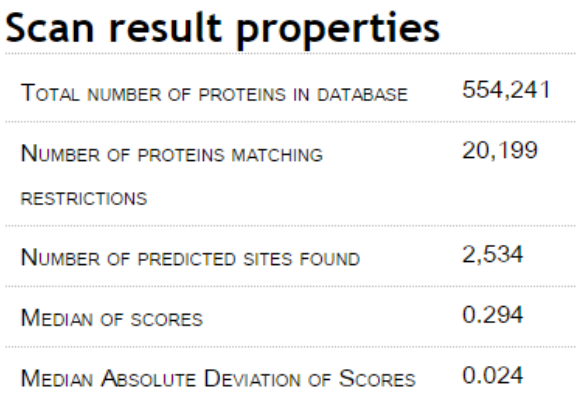


Figure 2.8: A *Scan result properties* block is displayed on the result page of the *Search a Sequence Database for Motifs* feature in *Scansite*. This particular section gives a first impression on the search results by displaying relevant numbers. If the number of proteins, that are in the database (in this case *UniProtKB*) is compared to the number after applying restrictions, the decrease of the amount of proteins to further process decreases outstandingly. In a next step, the proteins, that are actually relevant for scoring, because phosphorylation sites can be found, is even smaller. Following this step by step decrease gives an impression on how effectively the restriction parameters have been used before starting the search. The picture also indicates the quality of the scores, as their median and MAD are available.

3. **Predicted motif sites:** As in the protein scan, also the database offers a result table with multiple columns. The structure of the table however is slightly different.
   - **"Scan this Protein!":** Once a database search is run, it is possible to click on the link in this column to run a protein scan based on a particular protein, that is scored for a specific result entry.
   - **Score:** In the database search only the *Scansite* score is necessary, since any result is related to a single specific motif. Consequently, results can be compared directly without having to come up with a percentile.
   - **Accession:** Information in this column contains the identifier of the scored protein. Clicking on the identifier opens a new browser window, that contains all information about the protein. Usually the website of the original data source (e.g. *UniProtKB* by using default search settings) is the destination of the link in the *Scansite* table.

- **Protein Annotations** include alternative protein names, accession codes and further keywords. Basically a list of possible identifiers and important codes, that provide further information, are listed. During the development of *Scansite* 4 the formatting was improved. Instead of displaying the text in one block without a structured format using white spaces, its appearance has changed to show only the beginning per default. By clicking on the cell of the table, a well formatted description block pops up right below the result entry line.
- **Site** has the same purpose and structure as in the protein scan. The potentially phosphorylated amino acid together with its position in the protein sequence are displayed.
- **Sequence** is also comparable to the column of first essential *Scansite* feature result table. A 15 residue peptide of the protein sequence around a potential phosphorylation site is displayed. Also in the database search, the text formatting of the sequence was improved during the development of version 4 of the software application.
- **Previously Mapped Site:** Another new feature in *Scansite* 4 is the ability to reference previously mapped phosphorylation sites based on evidence data also in the database search. If the site is already known, there is an opportunity to look it up on its original source like the *PhosphoSitePlus* website, that is linked in every result entry with known sites.
- **Molecular Weight:** This value is related to the scored protein. Since *Scansite* 4 weight values are well formatted and rounded to two decimal places. Molecular weights are displayed in kilo Dalton (kDa). The weight of the protein is relevant, as methods in biology such as *mass spectrometry* highly depend on protein and peptide masses. Thus providing mass values in the result table can be very useful.
- **Isoelectric point:** The isoelectric point (pI) is also rounded to two decimal places. This value is an indicator for the behavior based on the pH-value of the area around the protein within a cell.

4. **Additional analyses:** The last part of the result page only offers to download all results, not just the displayed ones, to do further analysis independent of *Scansite*.

2.5.3 Find Sequence Match

The third out of seven features is the sequence match. By running this part of the program, a user is able to enter a sequence pattern with up to 15 amino acids. As for user defined motif insertions in previous features, it is possible to use wild card symbols next to single letter codes of the 20 naturally appearing amino acids. If required, multiple sequence patterns can be entered in the elements of the form. Except for the missing *sequence regular expression*, the restriction options of the sequence match are identical to those of the database search. Not only parts of the input mask are similar to the database search, also the structure of the result page is identical. The first block (Scan properties) includes an overview of the search settings and the final section (Additional analysis) offers the opportunity to download all results. Major differences are in the centrally located parts of the result page.

The *Scan result properties*, as shown in Figure 2.9, gives an overview regarding the number of proteins in the database as well as the amount that matches after considering given restrictions.

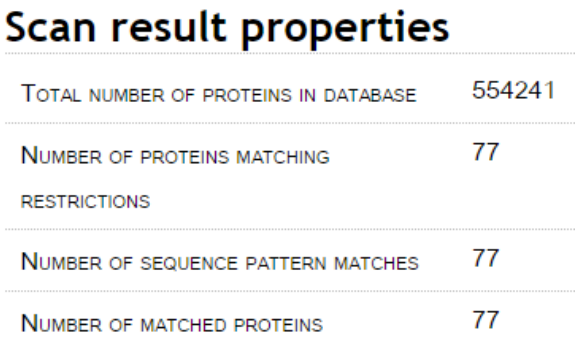


Figure 2.9: The *Scan result properties* block is displayed on the result page of the *Find Sequence Match* feature in *Scansite*. As in the database search, this particular section gives a first impression on the search results by displaying relevant numbers. Again the number of proteins in the target database (once more the default database *UniProtKB*) is displayed. Additionally feature specific values like the number of proteins, that match the feature and further restrictions are displayed. For this particular example, the sequence *S E E E D* – starting at position zero of the sequence – was used for the search. To further reduce the search space, the species restriction *homo sapiens* was also applied.

The result table with the actual sequence matches can be split in seven columns as defined below:

1. **Protein ID** shows the protein identifier, which – similar to the database search – is linked to a website, that shows further details about the matched protein. With the default settings the *UniProt Entry Name* is displayed and its related UniProt data page is referenced.

2. **Pattern:** This column only offers the opportunity to show the match, that was found within the protein sequence. Per mouse click on the cell of the table a box, that contains the protein sequence and the blue highlighted sequence match, is displayed.

3. **Protein Annotations:** This column is identical with the protein annotations in the database scan. Mostly alternative names, accession codes and keywords are listed within these cells.

4. **Molecular Weight:** As previously mentioned, a molecular weight value represents the mass of a protein in kDa and is an important measure, that is useful for multiple methods in biological research.

5. **Isoelectric Point:** As described above, the pI provides information about the behavior of a protein depending on the pH-value of its surrounding area.

6. **Motifs at expected sites:** Optionally phosphorylation sites can be indicated by checking the box below the text boxes for the sequence input. It only makes sense to check boxes, that are related to Serine, Threonine or Tyrosine as no phosphorylation information about other amino acids is available. Only if a phosphorylation site is indicated, the search checks for motif matches and calculates scores. If a match is available, this column displays the name of the matched motif, if there is any. The motif name is linked to a pop up box with more detailed information as visible in Figure 2.10.

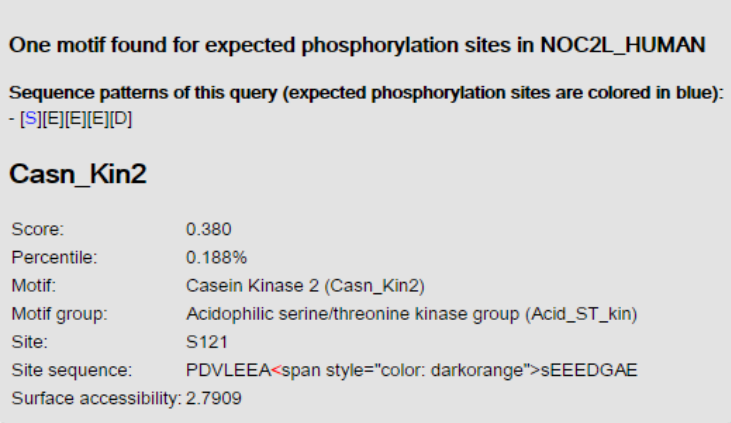


Figure 2.10: Detailed information about a motif match based on the result list of the *Find Sequence Match* feature is provided in a pop up box. After searching for the sequence *S E E E D*, whereby the Serine (S) is indicated as phosphorylation site, the species regular expression was set to *homo sapiens* to further reduce the search space. Within the result list, the scoring details of the match between the protein *NOCL_HUMAN* and the motif *Casein Kinase 2* are displayed. These details include most of the values, that are usually available in a result table of a protein scan.

7. **Evolutionary conservation:** If a motif is found at the expected site, it is able to run an ortholog scan, which is described below, by clicking on the blue *Scan orthologs* text.

2.5.4 Scan for Evolutionary Conserved Phosphorylation Sites

This part of the program is also referred to as *ortholog scan* and executes another protein sequence based search algorithm. Depending on similar protein sequence strings, common patterns can be found. This usually indicates mutations and other ways of modifying the genome. These changes happened to one particular genetic common species in the past. By slightly changing the gene pool, different species evolved out of a common root. By running this feature, a variety of modifications based on the previously mentioned common root can be compared to each other. The ortholog scan uses one protein of choice to search for other protein sequences, to find in most cases the same type of protein, which can be found in different species. Due to a common root, similarities between different species and thus transport mechanisms for drugs, that have been tested on animals, can be found.

Although this part of the application is primarily a sub-feature, that can be applied to results of a protein scan or a sequence match, it is available as a separate feature. For this sub-program, some input parameters are depending on other ones. Although it is recommended, to simply click on "Scan orthologs" in result tables of other *Scansite* sub-applications, parameters for the ortholog scan can be set as follows.

- **Orthology data source:** This first parameter decides which orthology database shoud be used. At this point, *SwissProt Orthology* and *NCBI HomoloGene* are available.
- **Choose protein by:** Currently it is only possible to choose proteins that are available in the *Scansite* database by using an identifier. The alternative option, which uses a user defined protein sequence, is currently disabled and it is not planned to re-enable the option.
- **Protein data source:** The protein database, that is used for picking the protein of choice completely depends on the orthology database. Identifiers for orthology database and protein database have to be identical to be able to run the ortholog scan. For instance, a valid combination is *SwissProt Orthology* and *UniProtKB*.
- **Protein / gene identifier:** As identifiers are already mentioned in the point above, it is absolutely essential to pick a protein identifier, that is recognized by both previously mentioned database types.
- **Search method:** The option with the biggest impact is most likely the search method. Depending on the choice, different input parameters are required as input.
  - **Sequence pattern(s)** If this (default) option is selected, a peptide sequence can be entered. The principle is the same as for entering the amino acid string in the sequence match input form.
  - **Motif groups** This alternative choice requires more detailed knowledge about the protein, that is used for the search. Next to selecting a motif group itself, that should be connected to the protein (e.g. listed in a protein scan result), also the position of the phosphorylation site has to be known and entered. Hereby only the position has be inserted as a numerical value (for instance 844 instead of S844).
- **Stringency:** This value is the same as for the protein scan. By choosing high (recommended and default), medium, low or minimum, the sensitivity of the search as well a s the overall quality of the results can be influenced.
- **Radius of sequence alignment:** This value is a parameter, that is relevant for the multiple sequence alignment only. By comparing sequences, only other proteins, that have a similar series of amino acids in a certain region, are supposed to have something in common. By defining how many positions before or after the position of the selected protein are relevant for a match, excluding randomly aligning sequences or refraining from matching wrong parts of a sequence can be ensured. Possible options for the radius are 10, 20, 40 (default) and 80.

To get a better impression of the parameters, Figure 2.11 provides an example for a correctly filled in input mask.

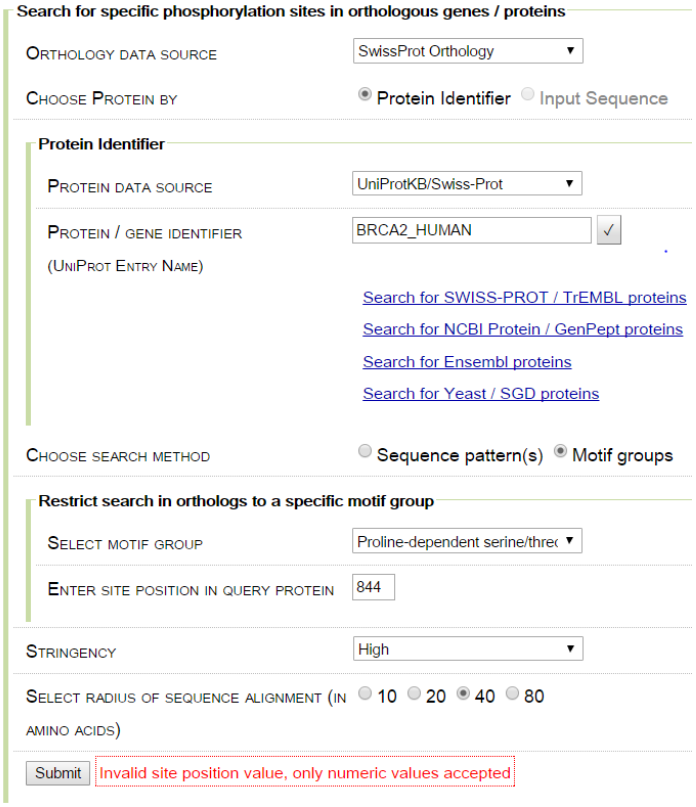


Figure 2.11: The input mask of an otholog scan uses the more complex *motif groups* option. This input is based on a protein result entry, which ensures the connection to the motif group *Proline-dependent serine/threonine kinase group*. A potentially Serine phosphorylation site is located on position 884. By executing the search, similarities to species like *Mus musculus* (mouse) or *Drosophila melanogaster* (fruit fly) are found.

2.5.5 Predict Localization

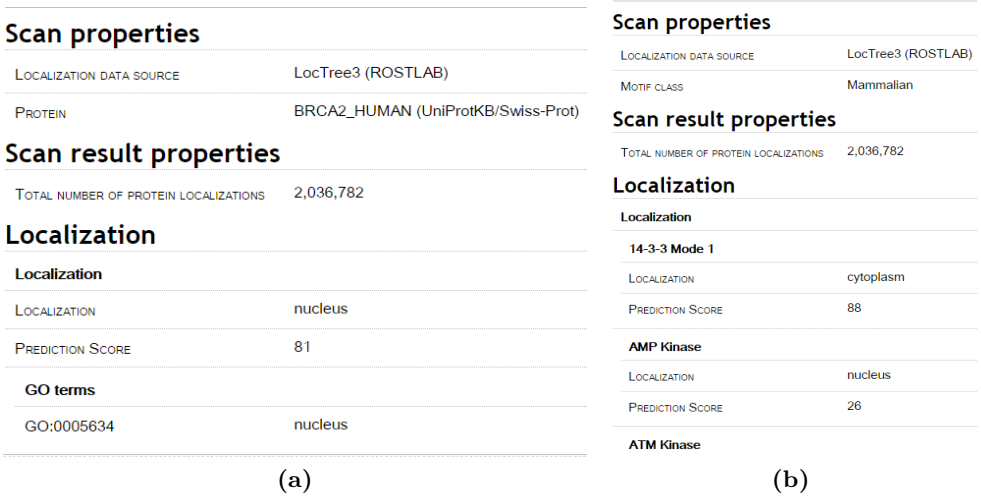
This part of the application can be seen as a sub-functionality of other major features. By using the only available localization data source *Loctree 3*, the subcellular location of a *UniProt* protein or of all motifs within a motif class (Mammalian or Yeast) can be selected as user input. Loctree uses *UniProt Entry Names* which means, that only databases with the same identifier type can be used for this feature, unless a new localization data source is going to be introduced in future. Consequently either the protein identifier (Figure 2.12a) or the motif class (Figure 2.12b) is effectively the currently only available input parameter. The result page in either case is split up into three areas as described below. The first two sections give an overview of input parameters and general information, whereas the last block provides actual data for the location within a cell. Results are structured as follows.

- **Scan properties:** This section shows which search option was used. The Localization data source is listed in either case. If a protein was used for the input, the protein name and its source are displayed. In case, a motif class was selected instead, either *Mammalian* or *Yeast* is shown in the second line of the section. Differences can be easily recognized in representative Figures (2.12).

- **Scan result properties** Information in this part always stays the same independent of search parameters. It covers the number of entries within the localization database table.
- **Localization** Whereas other features usually show large result tables, information in the localization prediction is just a relatively small list, which contains most relevant values (see Figure 2.12). Although the result page shows more details for a protein, it shows a more superficial list of the motif group search. The first option includes localization of the protein itself, a prediction score, which says how likely the first entry is to be true, and eventually the so called *GO terms*, which is also referred to as *Gene Ontology Evidence Codes*. Ashburner et al. [2] describe the purpose of their GO evidence codes as follows:

  > "The goal of the Gene Ontology Consortium is to produce a dynamic, controlled vocabulary that can be applied to all eukaryotes even as knowledge of gene and protein roles in cells is accumulating and changing. To this end, three independent ontologies [...] are being constructed: biological process, molecular function and cellular component."

  If the search parameter is a motif group instead of a protein, the result list looks slightly different. First of all, multiple entries (1 for every motif of the group) are listed. Second, the structure of a result entry is different. Such an entry can be described as a triplet. This includes the name of the motif, its predicted cellular location and finally a prediction score similar to the protein based localization search.



(a) **(b)**

Figure 2.12: The left image (2.12a) shows the localization prediction for a protein of choice. In this particular example *BRCA2_HUMAN* was used for the search on the left. The other picture on the right hand side (reffig:predictLocalizationMotifs) predicts the location of the phosphorylation site based on the motif kinases of the selected motif class. In this case the default value *Mammalian* was picked. By comparing the different output lists of the same feature, differences with respect to the outcome regarding search parameters are stressed.

2.5.6 Calculate Molecular Weight and pI

As the localization prediction, this feature can also be regarded as sub-functionality, that is separately available for the purpose of completeness. The molecular weight and iso-electric point calculator analyzes a protein sequence. It is possible to choose either a protein database and an appropriate identifier or a user defined protein by entering a name and a sequence in the input text forms. In both cases the maximum number of phosphorylation sites to be displayed also has to be entered. By default this value is set to five. The result has two different sections. The first part displays the protein sequence whereas the second one lists potential phosphorylation sites together with expected molecular weights and iso-electric points.

2.5.7 Calculate Amino Acid Composition

The last feature provides information about potential phosphorylation sites. Details about a relative amount of amino acids, that surround the site, are also available. As in the calculation feature of molecular weight and iso-electric point, a protein (sequence) of choice can be selected for analyzing. Results of the analysis are defined in 4 sections as described below.

1. **Choose center amino acid:** The first block is one of the basic interactive result elements. As phosphorylation sites in most cases are Serine, Threonine or Tyrosine, either one can be selected as central amino acid just by clicking on the radio button element. For extremely rare cases, a drop down menu also provides the option to select any other amino acid, that is available in the *Scansite* system.

2. **Ratio composition of all amino acids surrounding the centered amino acid**: Depending on the choice of the previous result input element, a table displays the relative amount for any amino acid around the actual site. The window, that surrounds the central position, contains – as in previous sections for the motifs – 7 residues. To better see a connection between interactive result elements, Figure 2.13 shows an example result table.

3. **Protein sequence and highlighted sites:** The last two sections of the result page are mostly depending on previous selections. Right below the table, that can be seen in Figure 2.13, the amino acid sequence is printed. Based on previous selections, certain residues are stressed using blue color instead of black. If domains can be found for the protein of choice, related regions in the sequence are also highlighted using background-color. As soon as the selection of the central amino acid is changed or if the user clicks on one of the cells of the table, highlighting for single residues change and the list of all highlighted sites at the bottom of the result page are adjusted accordingly.

Ratio composition of all amino acids surrounding Serine

Please click on one of the links in the table. The positions of the selected amino acid relative to the selected center (S/T/Y) will be highlighted. In case you are not sure about any of the amino acids' one-letter-codes, just move your mouse over these labels and a tooltip will present the actual name.

	A	R	N	D	C	E	Q	G	H	I	L	K	M	F	P	S	T	W	Y	V	U	O	s	t	y	r	k	l	
-7	0.04	0.03	0.06	0.06	0.02	0.08	0.05	0.03	0.03	0.05	0.10	0.08	0.02	0.03	0.06	0.10	0.07	0.00	0.02	0.06	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	-7
-6	0.06	0.01	0.10	0.06	0.02	0.07	0.04	0.06	0.04	0.05	0.06	0.05	0.00	0.02	0.06	0.13	0.07	0.02	0.02	0.04	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	-6
-5	0.03	0.03	0.05	0.07	0.02	0.09	0.04	0.03	0.03	0.06	0.09	0.10	0.02	0.05	0.04	0.11	0.08	0.01	0.02	0.04	0.00	0.00	0.01	0.00	0.01	0.00	0.00	0.00	-5
-4	0.04	0.03	0.06	0.04	0.02	0.09	0.05	0.03	0.03	0.07	0.07	0.09	0.03	0.06	0.05	0.12	0.02	0.01	0.03	0.06	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	-4
-3	0.04	0.04	0.04	0.05	0.02	0.09	0.04	0.04	0.02	0.05	0.14	0.08	0.02	0.03	0.04	0.09	0.06	0.01	0.02	0.06	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	-3
-2	0.04	0.02	0.07	0.04	0.02	0.07	0.05	0.03	0.03	0.04	0.08	0.10	0.01	0.02	0.06	0.13	0.07	0.00	0.03	0.07	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	-2
-1	0.06	0.01	0.08	0.05	0.03	0.06	0.01	0.04	0.03	0.06	0.09	0.09	0.01	0.07	0.04	0.08	0.09	0.01	0.02	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1
Ser	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	Ser
+1	0.05	0.04	0.07	0.07	0.03	0.10	0.02	0.07	0.03	0.04	0.10	0.10	0.01	0.03	0.03	0.08	0.05	0.00	0.02	0.05	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	+1
+2	0.04	0.02	0.08	0.04	0.02	0.07	0.04	0.03	0.04	0.06	0.09	0.11	0.02	0.05	0.04	0.13	0.03	0.01	0.02	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	+2
+3	0.06	0.04	0.06	0.05	0.04	0.08	0.06	0.04	0.02	0.04	0.08	0.08	0.01	0.04	0.04	0.09	0.08	0.01	0.02	0.06	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	+3
+4	0.05	0.02	0.07	0.04	0.03	0.07	0.03	0.04	0.04	0.05	0.08	0.09	0.01	0.06	0.03	0.12	0.06	0.00	0.02	0.08	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	+4
+5	0.04	0.04	0.07	0.07	0.02	0.08	0.04	0.03	0.03	0.07	0.06	0.12	0.00	0.04	0.04	0.11	0.06	0.01	0.03	0.05	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	+5
+6	0.03	0.03	0.07	0.05	0.04	0.09	0.05	0.06	0.02	0.05	0.06	0.09	0.01	0.03	0.04	0.13	0.06	0.01	0.02	0.05	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	+6
+7	0.06	0.02	0.09	0.04	0.02	0.06	0.04	0.04	0.03	0.06	0.09	0.08	0.02	0.04	0.03	0.10	0.07	0.01	0.02	0.08	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	+7
	A	R	N	D	C	E	Q	G	H	I	L	K	M	F	P	S	T	W	Y	V	U	O	s	t	y	r	k	l	

Figure 2.13: An example table shows parts of amino acid composition results for the protein *BRCA2_HUMAN* with Serine as its centered amino acid. Depending on the central amino acid, the table adjusts its relative values. As each single cell of the table is an interactive element, highlighting of the protein sequence below and other information is adjusted to the selection of a cell within the table.

Chapter 3

Design and Implementation

This chapter describes major changes, enhancements and upgrades regarding the most recent evolution of *Scansite*. While evolving *Scansite* from version 3 to 4, four pivotal aims were pursued. The first goal, that deals with improving the *Scansite* application itself, is the most fundamental one. However, software programs as well as other methods in almost any field of research always offer room for improvement because new information, technology or other reforms are available. As a consequence, the second aim focuses on future developers to get started as quickly and easily as possible. This includes certain processes, that are related to automation, and sophisticated as well as effortless deployment on the server environment. To make this possible, new implementations should work without issues and more importantly should be tested before deploying changes on the server. Thus the third aim is to make it possible to create a comparable development and testing environment, that is completely independent from the server itself. Obviously, there can still be issues, if software is tested on different operating systems (OS). Fortunately, differences in behavior, that are related to a specific different OS, are often well known and a solution is most likely easily accessible. The fourth and final aim of this thesis refers to security aspects. Certain steps and mechanisms are necessary to ensure, that valuable data can not or at least not easily be stolen. The security of particular information is highly relevant, as this data makes *Scansite* unique and special. Hence future funding and consequently future development and related research are connected to this last, but highly important goal.

In order to meet all those above mentioned objectives, developing software, setting up a work and test environment, adjusting settings, working out automation processes and many other things were necessary. The following sections of this chapter include tasks, that were fulfilled with at least one of the previously defined aims in mind. Details about how these tasks were implemented together with their purposes are defined later in this chapter.

3.1 Local *Scansite* Setup

The first necessary task was directed to the second aim. This included working out a development environment for developing *Scansite*. For this purpose multiple system, software and project requirements had to be met. Regarding the developer system,

the following software had to be installed and also needs to be installed on any future *Scansite* development computer.

- The latest *Java* Development Kit (JDK, currently Java version 8) is essential to be able to execute and test the program.
- *Apache Maven* as package and project management tool handles configuration specific aspects.
- An Integrated Development Environment (IDE) like *Eclipse* or *IntelliJ IDEA* is used to implement, handle and test source code.
- A local *MySQL* database, e.g. included in *XAMPP* is needed to access different types of data.
- To retrieve code and make changes, a version and revision control system such as *Git* is needed.
- In some cases packages and add ons for an IDE to fully support *Google Web Toolkit* are necessary.

Next to preparing the system and documenting the setup for future developers, *Scansite* – as a project – required multiple changes to be back in service on a new and more modern system. Since the project had not been worked on in years, many packages and libraries were outdated, and in some cases not even available anymore. Subsequently, some had to be completely replaced and the program had to be adjusted to new structures and interfaces. Next to updating and restoring previously working functionality, project configurations were fixed to enable logging. Consequently, info, warning and error logs could be used for bug fixing and further development. With the fixed project configurations, new developers can import existing code to their IDE, where building and compiling the program works out of the box by now. Of course IDE specific settings for running, building and debugging the software application still have to be set first by a developer. If no IDE is used, it is still possible to build the application by running command line instructions. *Apache Maven* is able to compile and build the project without configuring any settings.

3.2 Re-Enabling and Finishing Features

Besides general system and project related attributes, there were three parts in the code, that offered enough room for improvement to be heavily adjusted or even completely replaced. Those parts provided either previous functionality or code fragments, that were prepared to be extended and thus enabled in future. As all three functional blocks were supposed to have big impact on *Scansite*, tackling these issues was absolutely necessary.

3.2.1 InterProScan

The first of those three functional entities in the code is about executing a third party software called *InterProScan*. The *InterPro* software published by Jones et al. [19] is able to provide data about protein domains and also protein families. However, *InterProScan* is just the software, that is able to look up data in the *InterPro* database, that is freely accessible. By using this database, it is possible to find connections between proteins based on sequences [11].

The purpose of this third party software is primarily the option to display protein domains in a feature called *"Scan Protein for Motifs"* (see section 2.5.1). First of all, installing the previous version of *InterProScan* was a fairly complex task, as the *Scansite* target platform was a Linux system and additional software as well as multiple perl packages had to be installed. Despite installing all requirements, the application was still incompatible with latest packages and standards. As a consequence the upgrade from version 4 to *InterProScan 5* was required. A major issue was a missing *Linux* package, that was no longer available for downloading. Fortunately, the latest software release of *InterProScan* was written in another language and the package was no longer necessary. Instead of using perl, the *Scansite* program needed to be adjusted to call another *Java* application instead. Also input parameters were different. In place of a protein sequence, a FASTA formatted string was required in the new version. As a consequence, each sequence, that was prepared for the *InterProScan*, had to be extended by adding a FASTA header to its beginning. Next to the FASTA input parameter, also new program specific and partially mandatory parameters had to be introduced.

With respect to the third aim, that was defined at the beginning of this chapter, a test environment was required. It had to be separate and independent from the previous active *Scansite* server. Thus a test and development server was brought into service. On this machine, the latest *Ubuntu* version *Xenial Xerus* (16.04.2 LTS) was installed as operating system. Hence a Linux environment with the opportunity to install necessary server software and *InterProScan* versions was given. To be able to use *InterProScan* 5, only downloading the *Java* application together with the *InterPro* database was required. In order to use the third party software, it was necessary to define command line instructions. New commands had to be introduced, as the upgrade of the *InterPro* program required different input parameters. A new command was defined as follows.

```
java -Xms512M -Xmx2048M -jar interproscan-5.jar -i inputFile -o outputFile -appl
     Pfam -t p -f TSV -u ./
```

The `Xms` and `Xmx` commands define memory use and are not particularly relevant. These values were picked to provide a larger amount of memory for the *InterProScan* execution. `-i` and `-o` are identifiers, that have to be followed by a file name. These file names include the input file name with the FASTA formatted content and an output file name to write the results to. As the tool offers multiple different scans, which could consume a lot of computation time, only the *Pfam* application is used (as `-appl Pfam` indicates). By using `-t p` the input sequence type is defined. This is either *n* for DNA/RNA or *p* for protein sequences. `-f TSV` defines the output format. Hereby *TSV* means *tab separated file*. The last parameter `-u ./` defines the *InterProScan* working directory for temporary files. Since files are deleted after each run anyway, the *InterProScan 5* home directory was picked as workspace.

Per default *Scansite* is not allowed to access files outside the *Apache Tomcat* deployment folder. To be able to run *InterProScan*, access permissions for executing the third party software were granted and the tomcat user on the system. After the feature to scan for protein domains had been available, version dependent changes in the *InterPro* database had to be considered. Names of many domains were longer

than before in version 4, which resulted in overlaying names on the protein scan result page (see Figure 3.1). As cutting off the names would result in a loss of data, and editing the process of transforming the text of the domain image seemed to be complex and inefficient, a new table was introduced below the images of protein domains and surface accessibility. This table lists an entry for every single domain appearance in the picture. Even though the color code is not realized in the table, the full domain name, an IPR code and an alternative domain name are listed for every entry (see Figure 3.2). The order of the rows of the table is the same as the order of the domains with respect to their positions on the protein sequence. The IPR codes in the table link to the website of the *European Bioinformatics Institute*, which provides further information about the protein domains. To be able to receive IPR codes, an additional parameter `-iprlookup` was necessary. If this program argument is included in the command, the *InterProScan* software also includes a search for corresponding *InterPro* annotation, which yield the codes.

Figure 3.1: An extract of the protein domain plot is part of the *Scan Protein for Motifs* result page. *BRCA2_HUMAN* of the *UniProtKB* was used as protein of choice to search for. Except for enabling the *show predicted domains* feature, all other parameters were applied with their default settings. The extract makes clear, that *InterProScan5* domain names are relatively long. Consequently labels overlap. The order of the domains, the beginning and the end however, are clearly visible.

Full Domain Name	IPR Code	Alternative Name
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2 repeat	IPR002093	BRCA2 repeat
BRCA2, helical	IPR015252	Breast cancer type 2 susceptibility protein, helical domain
BRCA2, oligonucleotide/oligosaccharide-binding, domain 1	IPR015187	BRCA2, oligonucleotide/oligosaccharide-binding 1
Tower	IPR015205	Tower domain
BRCA2, oligonucleotide/oligosaccharide-binding, domain 3	IPR015188	BRCA2, oligonucleotide/oligosaccharide-binding 3

Figure 3.2: An example of a protein scan result page shows the newly introduced protein domain summary table. Settings for running the scan were the same as in Figure 3.1 to be able to establish a connection between visualization and listing. With the benefit of the format of the table, all domain names are completely visible and easily readable. Additionally the order in the table is the same as on the domain plot. Hence each and every protein domain can be identified in the image. On top every IPR code entry links to the *European Bioinformatics Institute* web page, which provides further information about the protein domains and families.

3.2.2 Public and Private Motifs

As mentioned in the Background chapter (section 2.3.2, point 6), preparations for distinguishing public and private motifs were available. Both, the motifs and the *Scansite* system had been prepared for this feature. During the development of *Scansite* 4 this part of the software was completed. As visible in Figure 3.3, additional motifs are available, if private access is granted. Private access is only available, if a user is logged in. Credentials for a *Scansite* account are currently limited and a registration is not possible without contacting a system administrator. Some of the not publicly accessible motifs are not yet published or still in development, which requires further experiments or confirmation.

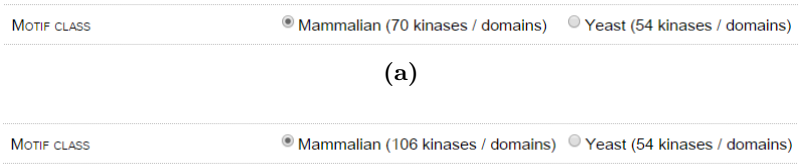


Figure 3.3: Once a user is logged in, next to regular 70 Mammalian motifs (see Figure 3.3a), additional 36 motifs are available for private access (see Figure 3.3b). Private motifs are reserved for further development and are most likely going to be publicly available after publishing them as part of a research paper.

3.2.3 Database Access Management

Previously a database management system had been available and *Scansite* itself was able to run, although database access related libraries caused trouble for logging in general. In addition, there were issues, when it came to intense use of *Scansite*. Especially for testing purposes or by providing *Scansite* functionality as a library caused errors. The old version, which used the *Maven* repository `tomcat-jdbc` from `org.apache.tomcat`, also contained certain logging libraries, that were not trivial to exclude. Having multiple logging libraries with different versions on top resulted in disabling logging in *Scansite* completely, unless correct settings specific to a respective IDE could be found. Nevertheless, logging did not work on the target platform, the *Apache Tomcat* server due to the broken project configuration.

Next to the rather trivial logging issue, establishing connections to the database was way more problematic. As described later in this chapter (section 3.12), tests were introduced and the *Scansite Web Service* was upgraded to the new version of the application. Both, tests and the web service, required the original *Scansite* application as a library, to be able to be executed. Once *Scansite* was included as a library, its functionality could as easily be accessed, as if it would have been part of the program. Consequently, multiple instances of a protein scan, a database search or essentially any other provided feature needed to be instantiated in order to use the features. Each time, one of the sub-applications was tested or run, a new temporary instance was created and each and every single one of those instances established its own connection pool with at least ten connections. If one considers heavy use of

the web service or multiple tests, a huge number of connections was the outcome. This number exceeded the maximum value for allowed connections to the database. *Scansite* did not close any of its connections by exiting the feature, because all connections were supposed to be reused. The only options to decrease the number of connections were either shutting down all connections manually or waiting for the database to close unused ones based on a timeout.

Showing this behavior, *Scansite* was not testable and further applications, that were supposed to use *Scansite* were not allowed to run more than 6 calls within the automatically configured connection closing time range of 30 seconds. Since this was a very critical state, the entire database access management was replaced. Essential changes are depicted in Figure 3.4, whereby the biggest change was switching from a connection pool for each feature to a single, unique and shared pool with per default 25 connections, that were all established as soon as the application was launched.

3.3 New Motif Structure

The most striking change in the upgrade from *Scansite* 3 to version 4 was introducing modified amino acids (primarily phosphorylation) in the kinase motifs. Being able to use modified residues allowed to stress phosphorylation sites, that depend on other previous or following modifications. Only modifications, that were supposed to be most common, were introduced to *Scansite* 4. Post-translational modifications (PTMs), that are now available in the application, are listed below.

- Phosphorylation of Serine
- Phosphorylation of Threonine
- Phosphorylation of Tyrosine
- Methylation of Arginine
- Acetylation of Lysine
- Methylation of Lysine

To be able to use these modified residues, several changes were necessary on two sides, the database and the *Scansite* code. For adjustments of the first part new columns in database tables had to be created. One new column per modified residue was needed to be able to store PSSM values respectively. Additionally all changes had to be transferred to the default *SQL* script, which contains create and insert statements for all basic *Scansite* tables. On the other end, the software itself had to be able to deal with the extensions of its data source. First, predefined amino acids in the *AminoAcid.java* class only included 22 amino acids, and thus were lacking modified residues. Those were completed by adding full names, four letter codes (1 letter for the modification and the typical 3 letter code of the AA), single letter codes, surface accessibility values and molecular weight. For all phosphorylated residues the lower case letter of the unmodified amino acid was selected as one letter code (s, t, y). The same was applied to methylated arginine (r). As there were two modifications for lysine, *k* was used for the more common modification, the acetylation and without any particular reason a lower case *L* was picked for its methylated form. Since Leucine is an AA, that is not supposed to be important for PTMs, potential future introductions of new modifications should not cause an issue regarding

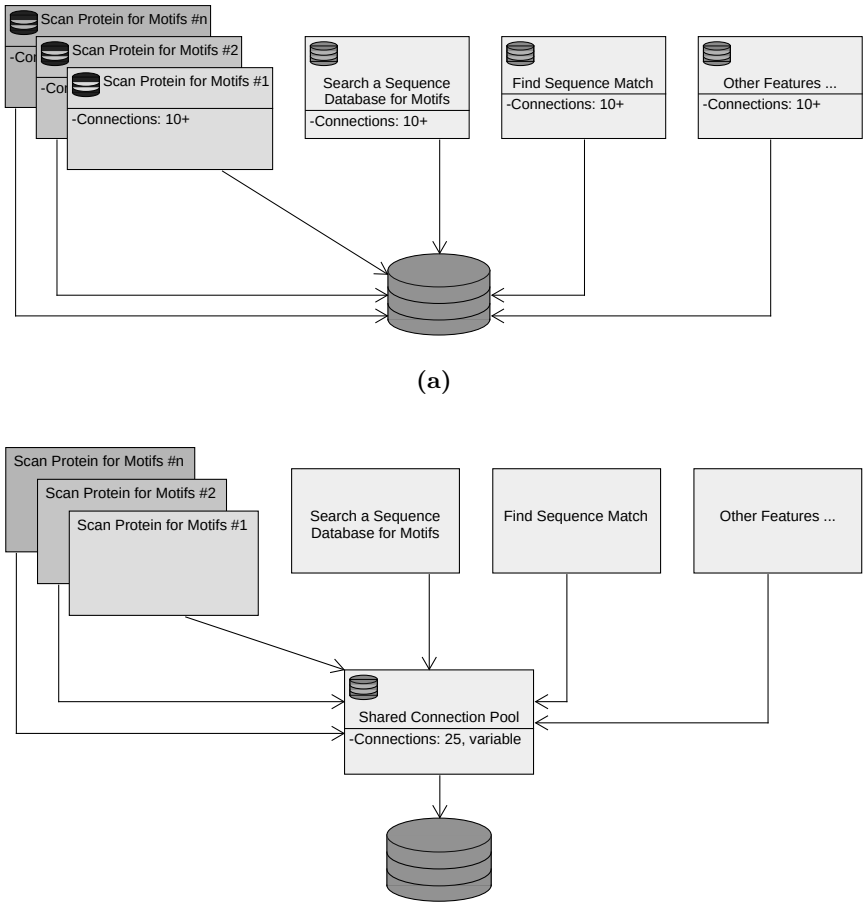


Figure 3.4: The old database access model (see Figure 3.4a) is compared to a new centrally managed approach (see Figure 3.4b). Whereas the old system is limited to the *Scansite* main application, the new database management is more robust. The old system, which established a reusable connection pool for every single feature, was not engineered to allow multiple instances of those features. Nonetheless, in library use (e.g. for testing), multiple instances of *Scansite* features were absolutely necessary. Since connections of terminating features had not been closed by *Scansite*, inactive and unusable connections were still open for about 30 seconds. This behavior easily caused an error, which indicated, that the maximum allowed number of database connections was reached or even exceeded. The new database handling in Figure 3.4b introduces an additional layer of abstraction. This layer includes a commonly shared connection pool, that is accessible to all features. Each new feature uses or reuses the existing connection pool instead of establishing new connections.

current choices for one letter codes. The masses of the modified proteins were calculated by adding values of amino acids, which were already in the *Scansite* system, to values of their modifications respectively. The modification numbers were provided by *Unimod*. How introducing the new residues in the code was actually implemented can be seen in Figure 3.5.

```
public enum AminoAcid implements IsSerializable {
    A("Alanine",         "Ala", 'A', 71.09310, 0.49, true),
    R("Arginine",        "Arg", 'R', 156.2018, 0.95, true),
    N("Asparagine",      "Asn", 'N', 114.1181, 0.81, true),
    D("Aspartic acid",   "Asp", 'D', 115.1029, 0.81, true),
    C("Cysteine",        "Cys", 'C', 103.1530, 0.26, true),
    E("Glutamic acid",   "Glu", 'E', 129.1298, 0.84, true),
    Q("Glutamine",       "Gln", 'Q', 128.1450, 0.84, true),
    G("Glycine",         "Gly", 'G', 57.06615, 0.48, true),
    H("Histidine",       "His", 'H', 137.1554, 0.66, true),
    I("Isoleucine",      "Ile", 'I', 113.1737, 0.34, true),
    L("Leucine",         "Leu", 'L', 113.1737, 0.40, true),
    K("Lysine",          "Lys", 'K', 128.1884, 0.97, true),
    M("Methionine",      "Met", 'M', 131.2069, 0.48, true),
    F("Phenylalanine",   "Phe", 'F', 147.1909, 0.42, true),
    P("Proline",         "Pro", 'P', 97.13095, 0.75, true),
    S("Serine",          "Ser", 'S', 87.09245, 0.65, true),
    T("Threonine",       "Thr", 'T', 101.1194, 0.70, true),
    W("Tryptophan",      "Trp", 'W', 186.2275, 0.51, true),
    Y("Tyrosine",        "Tyr", 'Y', 163.1903, 0.76, true),
    V("Valine",          "Val", 'V', 99.14685, 0.36, true),
    U("Selenocysteine", "Sec", 'U', 150.0490, 0.26, true),
    O("Pyrrolysine",     "Pyl", 'O', 237.3090, 0.97, true), // not in scansite2

    pS("Phospho Serine",         "pSer", 's', 167.0732, 0.65, true), // phosphorylation +79.9799
    pT("Phospho Threonine",      "pThr", 't', 181.0993, 0.70, true), // phosphorylation +79.9799
    pY("Phospho Tyrosine",       "pTyr", 'y', 243.1702, 0.76, true), // phosphorylation +79.9799
    mR("Methylated Arginine",    "mArg", 'r', 170.2284, 0.95, true), // methylation +14.0266
    aK("Acetylated Lysine",      "aLys", 'k', 170.2251, 0.97, true), // acetylation +42.0367
    mK("Methylated Lysine",      "mLys", 'l', 129.1295, 0.81, true), // methylation +14.0266
```

Figure 3.5: This Figure shows which amino acids are available in *Scansite* together with relevant values. These values include the full residue name, a three letter code (which is extended for modified residues), a one letter code, molecular weight, surface accessibility and eventually a boolean parameter, that defines whether an entry is an amino acid or a wildcard. Wildcard symbols such as *X, B, Z* and *J* as well as N- and C- terminal values are also stored in this class.

Unimod is a database, that provides weight information about protein modifications and is designed to support mass spectrometry methods such as *de novo* sequencing [6]. After enabling modified residues in the core of the system, it was important to adjust further parts of the application. As soon as modified residues were potentially available in the database and parts of the program, it was important to define how to use them for matches. On the one hand, there were no motifs with a preference for modifications at non-zero positions and on the other hand, protein sequences did not contain any modifications. Whereas introducing a new motif seemed rather trivial, checking protein sequences for modifications was a greater challenge. This challenge was tackled by indicating modifications within protein sequences. Although it sounds very trivial, it is computationally impossible to calculate all combinations or even permutations for modified and non-modified potential sites in real time. Consequently based on given evidence data, local *UniProtKB* protein sequences were modified. To indicate potential modifications, letters in the protein sequences were adjusted accordingly. As evidence data only includes phosphorylation, the remaining three modifications could not be considered for protein sequence changes. However, as soon as motifs with any of the 6 modifications are available in *Scansite*, scoring will treat modified and non-modified residues differently based on their different PSSM values. This is not limited to phosphorylation. If a motif does

not have values for modified residues, which is the case for all previously available motifs, default values are set. These defaults are the same numbers, that are representing non-modified residues. To save runtime, *Scansite* skips calculation of scores for modified AAs, if those have the same values as the non-modified residues.

At this point, modifications were supported in the database, modified residues were defined in the appropriate *Scansite* class and *UniProtKB* sequences indicated phosphorylation sites. However, actual *Scansite* features needed to be edited, as those did neither use nor support the new structure. Processing on the server side as well as displaying and highlighting modifications on the client side was implemented to eventually enable searches, that support modifications. The only remaining part to test and evaluate this new major feature was a motif, which had different values for modified and non-modified amino acids. To tackle this remaining task, a new private motif, which was determined by *OPLS* experiments, was inserted into the *Scansite* database.

3.4 Database Search Received *Previously Mapped Sites*

As mentioned in the Background chapter (section 2.3.3), the *Scan Protein for Motifs* feature offers one column in its results table, which shows *Previously Mapped Sites*. Again, these sites refer to earlier discoveries, which usually are proven with *wet lab* experiments. Hyperlinks to websites with more detailed information are given. Whereas this is a very useful feature for examining one particular protein with respect to multiple motifs, it is an outstanding opportunity to underline connections between proteins in a database search with phosphorylation site evidence. Based on *PhosphoSitePlus, Phospho ELM* and *Phosida* data, new connections between proteins and their phosphorylation pattern might be detected.

3.5 Result Filter for *Previously Mapped Sites*

To further focus on results, that show site evidence, a filter option was introduced. A check box, that was labeled "Show only previously mapped sites" was added to the two *Scansite* core features, the protein scan and the database search. If the check box is enabled, all results are filtered for evidence data. Only results, which actually include site evidence are displayed in the result table respectively. Since the database search has already had a filter for the number of results to be displayed, a two layered filter had to be applied. First, the results were filtered for evidence, and if there were still more results, than the "Output list size" allowed, a second filter based on the score was applied. If filters would have been applied in the wrong order, only results with evidence out of the best 50, 100, 200, 500, 1.000 or 2.000 matched results would have been displayed. This way there would have been a huge difference between the size of the result table and the output list size in most searches.

This feature is not just able to give a quick overview of all previously found phosphorylation sites, it is also able to underline the likelihood of *Scansite* results based on whether previously mapped sites are available or not. Having only results with site evidence also ensures to exclude possibly random results and might also indicate

new connections between proteins or motifs, that would not have been obvious, if hundreds of results would be in between or if the result would not have been displayed at all, as a score might not be as good as the ones from matches without related evidence data.

3.6 Motif Logo Rework

To make it easier to interpret a motif, the visual motif representation (motif logo) was enhanced. Even though the previous coloring was based on certain values and colored similar amino acids with a theoretically similar color, differences between colors were too big to recognize relations between AAs. Unfortunately, the old color system was often interpreted as random coloring for each amino acid. Thus and because of other reasons such as scaling, the appearance of the logos had to change. *Scansite* 3 used the number 21 for amino acids on the zero position also for generating the logos and set it in relation to the remaining values on different positions. The ratio between this arbitrary set value and real OPLS values does not contain any information at all. However, due to big differences in size compared to the central position, it was often hard to identify letters on any non-zero position. As a consequence all letters, that were not in the center were scaled up.

In order to realize the scaling process, a few steps were necessary. First of all it was necessary to check, whether the *magic number* 21 was used for the zero position. If this was not the case, no scaling was necessary. Any other way, in a second step, the algorithm calculated the sizes of all other columns and checked, which one of the non-central columns was the biggest one. In a third step, the ratio between the height of the zero column and the biggest of the other ones was calculated. Eventually this number (central size divided by biggest non central column size) was used as a scaling factor, that was multiplied to every non-zero column amino acid letter size.

Next to the different ratios of the motifs, Selenocysteine (U) and Pyrrolysine (O) were removed from the motif logos, if they had default values assigned (those of Cysteine and Lysine). If these super rare amino acids were not part of any OPLS experiments, including U and O would only confuse people and the logos would simply not be correct. Huge differences, which are results of the rework, can be recognized by comparing the old logo of the motif *14-3-3 Mode 1* from the Background chapter in Figure 2.3 to its improvement in Figure 3.6.

Another innovation for the motif logos, that is visible in Figure 3.6, is a legend. This legend describes the new color codes and provides information about meanings of all colors used. Descriptions are split in three groups, which include (i) general color codes valid for all motifs including new modified residues, (ii) aromatic/aliphatic coloring, and (iii) a polar/non-polar scheme. At the moment regarding modifications only motifs with phosphorylation sites are available in *Scansite*. Hence colors and descriptions for the remaining three modifications in the system are not available yet. For the general group, mostly commonly selected colors were used. Acidic amino acids are red whereas basic AAs are blue. Specific to *Scansite* residues on the central position, which are special, are the only ones in black. To acknowledge feedback on *Scansite* during the testing stage and in order to stress specific AAs, pink

was selected for phosphorylated residues. As Proline plays a special role in protein sequences, it is the only green amino acid in the logo. Next to fundamental coloring, state specific coloring is also available. In *Scansite* 4 a new button, which is labeled *"Switch to polar & non-polar coloring"* per default, was introduced. Once this button is clicked, the default colors for aromatic and aliphatic amino acids (see Figure 3.6) are replaced by polar and non-polar coloring (see Figure 3.7). The label on the button also changes to *"Switch to aromatic & aliphatic coloring"* to be able to go back to the default view. This way two different coloring schemes are available and either one can be selected just by clicking the button. The default with the aromatic/aliphatic color palette includes yellow (for small AAs), brown (for hydrophobic ones), purple (for aromatics) and orange (for the remaining ones) as visible in Figure 3.6. If the alternative coloring is selected, only purple (polar) and orange (non-polar) are available next to previously defined general default colors as shown in Figure 3.7.

Another new sub-feature is saving actual values for amino acids on the zero position in addition to the value 21. So far these real values are only used for the logos as depicted in Figure 3.8. However, this way real ratios to different columns of the logo are given and the relation of the sizes to each other actually have meaning with respect to biological methodology. The greatest benefit regarding the use of real values is certainly the ability to depict and compare actual ratios in between amino acid values of the central position.

3.7 Local Database Setup

To further pursue aims 2 and 3, which try to work independent of the server system and provide defined processes for future developers, a standardized way to access a database was needed. Whereas a *Scansite* database server had been available in the past, a local database setup was required. The previous database server was still actively in use by *Scansite* 3 and thus was refrained from being used as development database. To be independent of an existing database on another computer, and also to be able to work offline, a local setup was an excellent solution. This setup together with its documentation was worked out to support future *Scansite* developers to get started quickly and easily without being able to break the current system.

As a local setup should be platform independent, *XAMPP* – which works on *Windows*, *Linux* and *OS X* – was the tool of choice to setup a *MySQL* database locally. Reasons for using this particular program were a straight forward and easy installation, automatic configuration, an opportunity to start and stop the database per mouse click (tested on *Windows* only) and the built in *PHP My Admin* feature. *PHP My Admin* is a web interface, which can be used to access the database without writing commands. It is particularly handy, if a quick overview for the entire database or some specific tables is needed. Next to the installation, the database had to be prepared for being accessed by applications like *Scansite* or a database test application, which both used predefined credentials. To enable automatic access, a user had to be created, privileges and usage had to be granted, and finally the local database had to be setup with basic tables. Subsequently it was filled with information, that was either predefined like user accounts, or provided by public databases. Eventually local motifs and phosphorylation site evidence data had to be added.

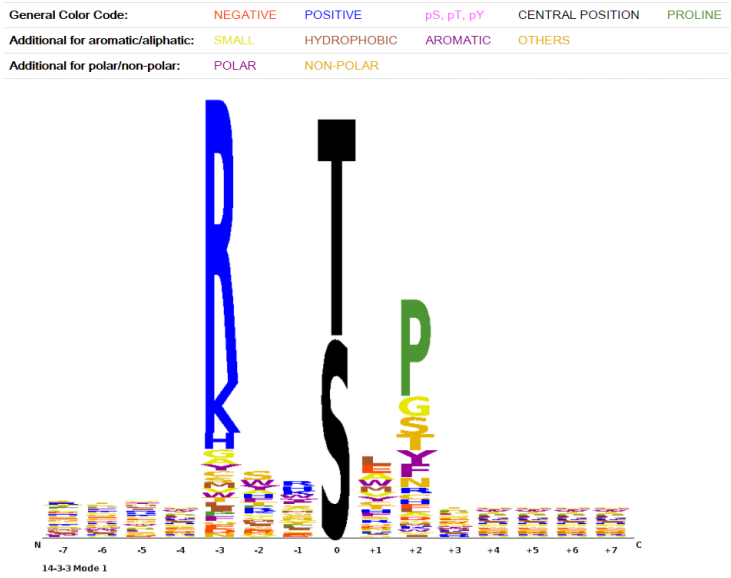


Figure 3.6: An example for a new *Scansite* 4 motif logo shows new scaling and color codes. To be able to compare it to an old logo in Figure 2.3, *14-3-3 Mode 1* was also selected as the motif of choice to display. By comparing those two motifs, it is easily possible to recognize, that non zero position residues are scaled up. Moreover the color scheme changed and a color description was added. The new coloring has less variety and focuses on distinguishing only most important attributes.

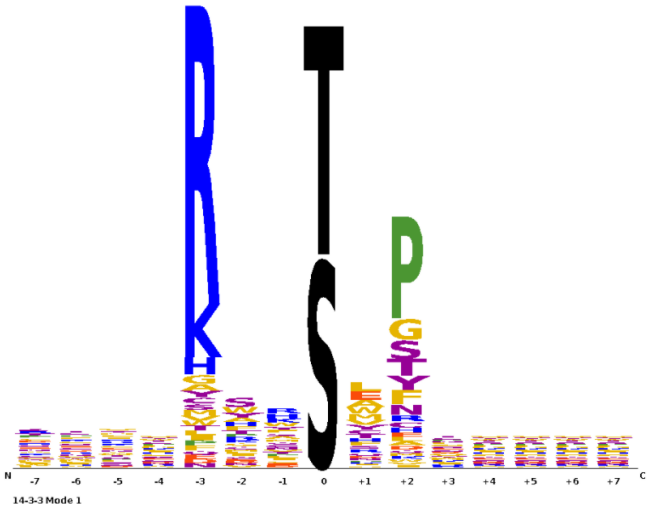


Figure 3.7: An alternative image for the new *Scansite* 4 motif logo, which can be retrieved by clicking a button, is depicted. Once again *14-3-3 Mode 1* was selected to be displayed. Compared to Figure 3.6, which displays per default the color palette for aromatic and aliphatic amino acids, the alternative image focuses on polar and non-polar coloring.

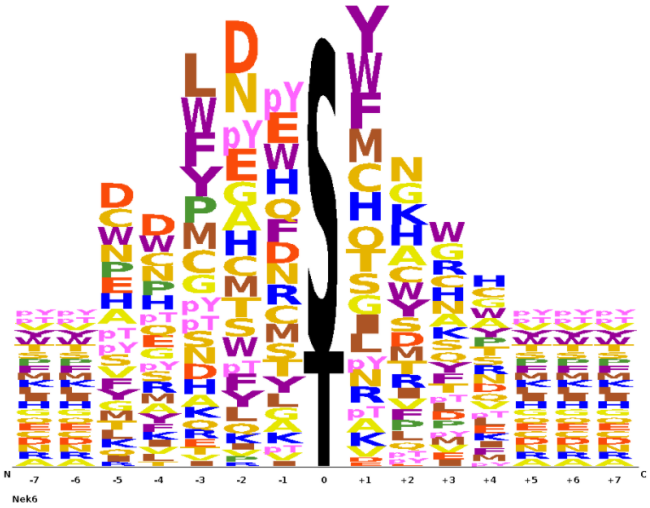


Figure 3.8: To demonstrate another innovation, a motif logo for a new private motif called *Nek6* is shown. The default image with the aromatic/aliphatic coloring is depicted. This picture particularly point out, that (i) new modified residues are available and visible in *Scansite* (pY, pT and in other motifs also pS) and (ii) real values for new motifs can be applied to the logos, which includes correct ratios and affinities in between the letters of the zero position.

To be able to do so, changes in *Scansite* were also required. Links for sources had to be updated and the database mirror mechanism had to be edited. Previously it was tried to process data from up to four publicly available databases simultaneously. As this routine did not perform well, it had to be changed. In fact, these parallel processes blocked each other. Whereas downloading multiple files at once may not have been a big issue, processing available data definitely was. The scheduler of the database, which was in charge of managing information of the database, had to switch between multiple insert queues. Subsequently, inserting data took much more time, than processing databases one by one. Additionally, the parallel approach violated constraints and caused errors. As a consequence, the process of inserting database content was adjusted to run in serial, one database after another. As the database setup still took hours, additional info logs were added to display infos such as download progress or the name of every database on which *Scansite* was working.

One more issue were error messages during setting up the database. A major reason for errors were *ALTER TABLE SQL* statements. Since the setup tried to include installing a new database as well as updating an existing one, commands were not specified to one particular task. To avoid errors at least on the developer system, that used a *XAMPP MySQL* database, afore mentioned statements were extended. In order to cushion an error because of non-existing tables, a check was added. This particular check looked up, whether a certain table existed and consequently decided to execute or to skip the statement. Subsequently, trying to rename a table, that does not exist, is not tried anymore. Unfortunately, the *SQL* check was not compatible with the *Linux MySQL* server. Thus it was only applied on *Windows* systems. On a server environment a setup was still able to run. The only difference were logging error outputs, which showed, that non-existing tables could not have been renamed.

3.8 Database Insertion Manager

Unfortunately, setting up a database did not work out of the box, neither locally nor on the target platform. To further evolve *Scansite*, a reliable mechanism for setting up and updating a database was essential. Without a new reliable database, testing different parts of the application would not have been possible. In addition, *Scansite* would be unable to operate, once the database server would break down. Such a database setup mechanism had to be able to fulfill at least the following jobs.

- Delete all temporary files from previous runs of the setup if there are any before actually launching the setup (e.g. aborted database setup run)
- Delete all temporary files after finishing the setup to use minimum disc space.
- Replace outdated data, which was mirrored earlier
- Setup a database from scratch
- Download information from public databases and process the files to be able to mirror those databases into the *Scansite MySQL* database
- Process and insert motifs
- Handle evidence data and additional (future) tasks easily

While evolving *Scansite* 3 to version 4, database related functionality was very limited and partially deprecated. As displayed in an overview in Figure 3.9, classes called *RunUpdater*, *RunEvidenceInserter* and *MotifInserter* were available. However, these classes were not tested and some sources were not available anymore. It was necessary to adjust those classes in a way, that allowed downloads using different protocols including HTTP, HTTPS and FTP. On top of that it was important to delete all temporary files, as downloads were only started, if no temporary files were available to process. In case temporary files existed, a not tested and not working update mechanism was triggered instead.

Another issue was the process, that was required to setup the database environment without having an IDE. To be able to setup the system with a minimal tool set, it was required to generate JAR files on the development machine. Once those files were available, they had to be copied to the target system to be executed one by one by typing in commands in the *Linux* terminal together with all required parameters. This was another reason for changing the setup routine in order to create a single central mechanism, which was supposed to be in charge of the entire setup in future. To create a reliable setup, methods of those three classes were put together in a management class called *RunDatabaseInsertionManager*. This common entity made sure, a clean environment was provided right before mirroring databases and inserting data. The insertion manager was implemented to drop the old database to create a new one as well as predefined tables, and to insert constant data such as *Scansite* users or news. Additionally all temporary files had to be deleted before and after running the setup. This way disc space was freed as soon as possible and the application was still reliable, even if a previous setup had been aborted. Once a database reset had been done, *RunUpdater*, *RunMotifInserter* and *RunEvidenceInserter* used to run in this particular order. Due to extensions and innovations within the program, new classes were introduced and a new order was necessary. As depicted in Figure 3.9, the database manager in its current stage calls its classes in the following order.

1. **RunUpdater:** Downloads data from public databases (see Figure 3.9), extracts and processes the information to insert it into the *Scansite MySQL* database.

2. **RunEvidenceInserter:** Right after mirroring public databases, evidence data is processed and also inserted into the database.

3. **RunSequenceModifier:** This new class is responsible for highlighting parts of *UniProtKB* protein sequences, that are connected to site evidence with respect to phosphorylation data of the previous step. Every expected phosphorylation site was marked by replacing letters in the amino acid sequence by their lower case representation. Each marked position indicates a phosphorylation site, which depends on further phosphorylation sites, around the site of interest.

4. **RunMotifInserter:** After preparing the sequences, all motifs have to be inserted. In this step also histograms and percentiles are generated. Since this step usually requires the biggest amount of time, it is executed at the end of the setup routine.

5. **RunMotifCentralPositionValueInserter:** Once the setup including all previous classes is complete (particularly inserting motifs), it is possible to insert real values for the central positions of the motifs to display correct values, as previously mentioned in the motif logo section (see 3.6). Executing methods of this class is optional, as the system does not depend on exact values for the logo representation. If no value is available, logos are generated with their default values, which means the zero position is set to 21 instead. Nonetheless this class is recommended to be executed as it does not require much time to be executed and its benefit can be very useful.

Because the new setup deletes any files, that are downloaded after finishing its execution, it can also serve as update mechanism. The setup is currently supposed to deal with a new system. Consequently, updating is not very sophisticated. The current database is completely deleted and a new one is setup and filled with more recent publicly available data. Motifs and evidence data are fixed and not subject to updates. Even though it takes a couple of hours to finish, the setup allows to maintain the system. This setup turned out to be especially useful for testing purposes. As a local setup does not require all databases, the ones to mirror could be selected in a configuration file and once requirements or the focus on current development changed, a different database was easily setup by changing the configuration and executing the *RunDatabaseInsertionManager* once more.

3.9 *In Silico* Motifs

To support future science with references and to underline correctness of *Scansite* input data, *in silico* motifs were generated using *PhosphoSitePlus* phosphorylation

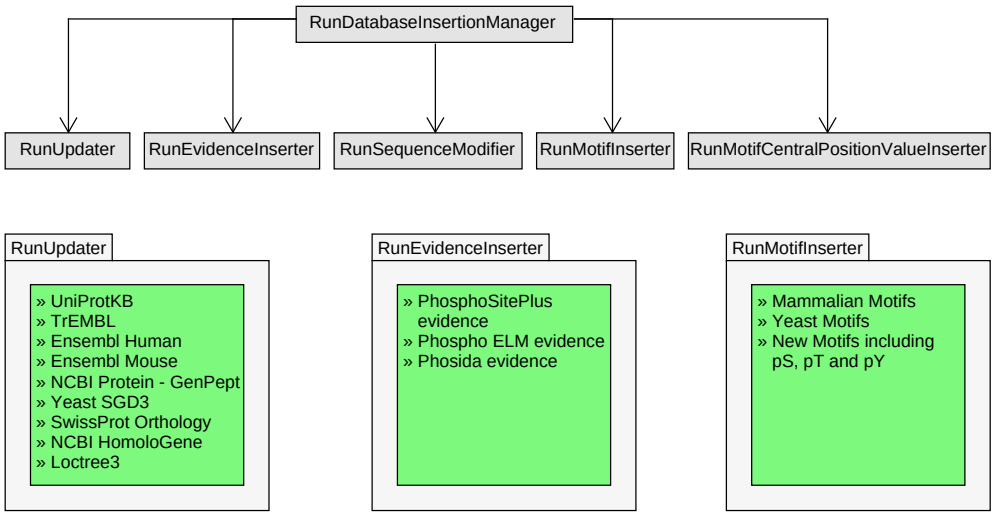


Figure 3.9: On the one hand an overview of all classes in the *Scansite* code, that are related to the database setup, is given and on the other hand more details about the already in *Scansite* 3 available classes are depicted right below. The three lower elements show, which databases are mirrored by executing the *RunUpdater*, which evidence data is processed and inserted by the *RunEvidenceInserter*, and what different motifs are available in the system. This includes both, public and private motifs. The upper part of the image makes clear, that the *RunDatabaseInsertionManager* handles the entire setup by using functionality of every single class in the same order as depicted. Setting up the system starts with a database reset followed by the *RunUpdater* and is technically finished after completing the *RunMotifCentralPositionValueInserter*.

substrates. Since those generated motifs do not necessarily have biologically correct and relevant values, although this could very well occasionally be the case, all generated motifs are restricted to private access. To be able to calculate the position specific scoring matrices (PSSMs), a number of steps were required. Before processing data was possible using *Python* scripts, some preprocessing steps were necessary.

Preparations included a search for phosphorylation substrates on the *PhosphoSitePlus* website. During the search for every kinase, that was related to an existing *Mammalian* motif, each occurrence of a substrate page was downloaded. As the default download format was *Excel Binary File Format* (.xls), each file was opened manually to be saved in a more modern format (.xlsx). After this step, a open source *Python* library called *openpyxl* (https://openpyxl.readthedocs.io/en/default/) was used to read the *Excel* input files. For each Excel file, sequences of phosphorylation sites were parsed, and for each motif a list of sites was generated. To be able to distinguish each list, the protein identifier of the appropriate kinase was added to each list as a key. In following, data was processed in multiple steps. First, a position specific weight matrix (PSWM) was created for all valid entries of each list separately. Valid entries were list elements, that were not N- or C-terminal (represented by a "_" symbol) and did not contain any non-accepted symbols for modified amino acids. Allowed symbols for the list of valid modifications were defined as listed.

- One letter codes of the 20 common natural amino acids
- Selenocysteine (U) and Pyrrolysine (O)
- Wildcards: B, X, Z, J
- Modified (most likely phosphorylated) Serine, Threonine, Tyrosine: s, t, y
- Modified (most likely methylated) Arginine: r
- Modified (probably in most cases acetylated) Lysine: k

All strings, that contained a non-accepted symbol were omitted for further calculations. For the remaining sequences, which was the majority, a matrix with the absolute frequency with respect to each amino acid symbol and to its position in the sequence was generated. In the next step the number of residues per position which is the same as the number of sequences, becomes important. By dividing each entry in the matrix by this relevant number, the PSWM is the outcome. To get close to real PSSMs, that are provided by OPLS experiments, a normalization step was essential. By multiplying the PSWM with a normalization factor, a PSSM for each kinase was generated. Every matrix was saved to a specifically formatted file respectively. This particular format includes the one letter codes for each included residue in the first line followed by all PSSM values in the following 15 lines. Each of the following lines includes values for a designated motif position.

3.10 Secure Traffic

In *Scansite* 3 there was the possibility to login to be able to manage motifs. It was possible to insert new ones, to edit and delete existing ones and with administrative privileges, it was even able to modify, add or delete user accounts. Messages between client and server application were not encrypted and basically any person, who would have tracked traffic, could have been able to get valid credentials to login. This could have caused serious issues like stealing motif data or deleting them on purpose to break the system. To prevent *Scansite* 4 from being hacked and to make it way harder to steal data, a MIT signed SSL certificate, specifically designed for the domain *scansite4.mit.edu*, was put on the server to run traffic encrypted on the web browser via *HTTPS* protocol. Having interactions between server and web browser encrypted is way more important than in *Scansite* 3, as the public/private feature grants access to additional motifs. As most private motifs are still subject to active research, data is more valuable and requires additional protection. Also more user accounts are available and use of *Scansite* is expected to further increase after publishing version 4.

To enable encrypted traffic, multiple steps were necessary. First of all a private key had been required on which the certificate was based later on. Following the guide on the *MIT Information Systems & Technology* website (http://kb.mit.edu/confluence/display/istcontrib/Obtaining+an+SSL+certificate+for+a+web+server), a private key for *Scansite* was generated by executing the following command in a *Linux* terminal on the *Scansite* server:

```
openssl genrsa -out/etc/ssl/scansite/scansite4.mit.edu.key 2048
```

In this command `openssl` is the *Linux* application that is executed. `genrsa` hereby represents the generation of a *RSA* key. *RSA* is a so called cryptosystem, which is

widely used to transmit data securely [26]. The `-out/etc/ssl/scansite/scansite4.mit.edu.key` part of the command defines the output file name and the directory, where the private key will be stored. At the end of the command, a number (in this case `2048`) defines the size of the key to generate. The unit of this value is in bits and if no value is given a default of 512 is applied. As key and certificate are stored on the file path *"/etc/ssl/scansite/"*, the command for generating the certificate request file could be generated with the command below:

```
openssl req -new -key /etc/ssl/scansite/scansite4.mit.edu.key -out /etc/ssl/
  scansite/scansite4-2017.req
```

Although this command uses the same tool, a different process is started. Here, a request (`req`) for a new (`-new`) SSL certificate is generated. This request is based on the previously created private key (`-key /etc/ssl/scansite/scansite4.mit.edu.key`). As before, also directory and file name for the output (`-out /etc/ssl/scansite/scansite4-2017.req`) is required in this command.

After entering required information for the certificate request, *MIT* certificate services was requested to generate a signed certificate. Once a SSL certificate was available for *Scansite* 4, it was copied into the folder, that already contained the private SSL key. In addition an intermediate certificate was also copied to the `/etc/ssl/scansite/` directory. Once all files were available for secure traffic, server settings had to be configured. Instead of directly accessing *Apache Tomcat*, which hosts the *Scansite* application itself, traffic is now directed to an *Apache* web server, which is better designed to deal with encrypted traffic. The web server then forwards all requests in plain text internally to the *Tomcat* server.

3.11 Automatic Server Deployment

To drastically make future *Scansite* development easier and faster, a deployment pipeline was implemented. This setup includes installing any software, that is required for a *Linux system*, to operate as a server for *Scansite*. Required software for hosting the web application included primarily the following elements:

- **cURL:** This part of the setup is optional. *cURL* provides an opportunity to download various content from the web by typing in `curl` followed by a web link. As *wget* is installed per default on *Ubuntu*, both tools can potentially be used to indicate downloads, and installing *cURL* is not necessary. It is useful to have it on the system, especially for downloading multiple files of a single source. In such a particular case it is possible to use wildcards in the download link. In general, *cURL* was required for preparing the previous *InterProScan* version. As the new *Interpro Scan 5* is written in *Java*, *cURL* becomes an optional part for the installation. At this point it is still included due to previously mentioned benefits. Another reason for keeping it is a quick and easy installation and a low use of disc space.
- **Java (JRE & JDK):** *Java* is a core requirement for *Scansite*. As the application as well as *Interpro Scan 5* are both implemented in this programming language, the *Java Runtime Environment* (JRE) is absolutely necessary. The JRE can be seen as a platform on which all *Java* applications are able to run.

Additionally, also the *Java Development Kit* (JDK), which is necessary for building *Java* programs, is installed for running build tools such as *Apache Maven* and for potential future server side development and tests.

- **Apache Tomcat Server:** Similar to *Java* also the *Tomcat* server is a very essential requirement for deploying *Scansite*. Actually the *Java Web Archive* (WAR file), which is generated by building the *Scansite* project, is automatically deployed on this server. The only necessary step is copying the WAR file to the *"WEBAPPS"* folder, that is related to *Tomcat*.
- **Integrated development environment (IDE):** Another optional part of the setup is an IDE. Per default *IntelliJ IDEA Ultimate* is installed on the system in case server side development is required in future. Of course other IDEs like *Eclipse* can also manually be installed and *IntelliJ* can easily be removed by deleting the folder, that contains the software.
- **Apache Maven** needs to be installed, to be able to build the *Scansite* project. This tool manages all third party dependencies, downloads relevant sources, runs tests and finally builds the application to create above mentioned WAR files and also *Java Archives* (JARs) for running the database setup, that is prepared in the *RunDatabaseInsertionManager* class.
- **Git:** The version control system is used as storage for *Scansite* 4 and its source code respectively. However, it is necessary to install the client part of the application to be able to access the code, which is stored in a repository. All approved changes are immediately available on the web, as usually only working and tested innovations are committed to the *Git* system. Once the changes are public, it is easily able to download the project together with all available code using the *Git client*.
- **Apache Web Server:** A server environment, that is at least as important as the *Tomcat* server is the *Apache Web Server*. Major traffic management is handled by this web server and the biggest amount of server side configuration is required for this tool too. Encrypted traffic is able due to SSL configuration of the web server and unencrypted interaction to the *Tomcat* server is run internally on the *Scansite* server computer. On top, the default HTTP web access on port 80 is redirected to encrypted messaging.
- **MySQL server:** The database server setup includes installing the database itself and launching the service. On top of that, also *Scansite* specific configurations are required. The database user management is adjusted by adding a new user. Subsequently, rights are granted and a specific command is executed. This allows to use the database freely through a single defined valid user.
- **SSH Server:** As a server is usually not connected to a monitor or peripherals such as mouse or keyboard, access is not possible by default. Whereas this is a great condition with respect to server security, sometimes access to the computer is required to run database updates or to install a new *Scansite* version. Also installing bug fixes of existing features next to updating the operating system is an important point. Consequently, a SSH server is installed to provide remote access to the *Scansite* server. Various SSH clients are able to connect to the server by entering valid credentials.

- **InterProScan** is a *Java* based third party application, that only needs to be downloaded. Since a JRE is installed, it is possible to execute the application without any further requirements. It is also important to set user privileges accordingly. At least one has to make sure, that the *tomcat* user, which represents access for applications on the *Apache Tomcat* server, is allowed to run *Interpro Scan 5*.

As installing and configuring the above described software requires the same steps each and every time, an automatic routine was implemented. *Bash* scripts, which partially depend on each other, provide a fully automated setup to prepare the system. As version numbers of some applications change sometimes, such specific values are saved in configuration files. Subsequently, configurations can be saved independently of the actual scripts. Thus a person who just wants to run the scripts with currently working versions does not have to understand the scripts. Just version numbers in provided configuration files have to be updated. Once all versions are correct, it is possible to run the entire setup just by running the main script. Depending whether one is using a test server or the actual *Scansite* server, a different script should be executed. In case one is working on the publicly accessible *Scansite* 4 server, one should use the following command to launch the script sequence.

```
sudo /home/scansite/Documents/shScriptsV2/setup_scansite.sh
```

In this case it is expected to copy configuration relevant files and the folder, which contains the scripts, into the Documents directory of the default user *scansite*. If the system is just a test system, that does not have the necessary address configuration to work together with the domain, another script has to be used instead. The setup script called above has set default values, that are connected to the real server. On a test computer, one does not necessarily need the *scansite4.mit.edu* domain to test functionality. As a consequence an alternative script called *main.sh* is able to override the default settings by reading the IP address of one of the computers network interfaces. To do so, one has to replace the predefined network interface name (enp4s0) with the one of the test computer. To find the name of the interface, one can run the `ifconfig` command in a Linux terminal. Of course one could just use the IP address, that is provided by the very same command. However, in case the test computer is connected to a DHCP server[1], a different address might be available if the computer is shut down in the meantime or if a setup is run multiple times due to testing new additional installations or configurations, that might damage the system. To put everything in a nutshell, one can adjust the main.sh script to any preferences. It is possible to do both, to configure an IP address or a network interface. After fully configuring the main script, the call looks similar to the one of the setup script. As shown below, the only replacement is the name of the script.

```
sudo /home/scansite/Documents/shScriptsV2/main.sh
```

Independent of which script is run, the setup itself is the same. Although user input has been minimized, some interactions with the setup routine are still required. For setting up the system user interaction is necessary as defined in the following points.

- **Root privilege confirmation:** To be allowed to run the script with the `sudo` prefix (super user do), the password for the user *scansite* has to be entered.

[1]Being connected to a DHCP server means, that addresses are received automatically and an address of a specific computer might change over time.

- **Database root user:** For the installation of the *MySQL* database, a password to the database root user is requested. One has to enter a password for this one twice, once to set it and once to confirm. To keep the system consistent, it is recommended to use the same password as for the *Scansite* user.
- **Database related configurations** are required. Those include changing the root password, enabling a password check (verifies strength of passwords), discarding test users, test databases and other trivial things, that are not required anymore. Whereas the database password, that had just been set, does not need a change, and the password check is not really necessary, all following options to get rid of superfluous data are very welcome.
- **Apache configurations:** In this step default values make sure to use the correct server (*apache2*) and to setup *MySQL* database configurations from scratch.
- **Finishing the setup:** To finish the setup, the database root user password is requested three more times, to setup the *Scansite* specific database as well as a user with all necessary privileges automatically.

As mentioned before, the `setup_scansite` script can be called directly from the *Scansite* server due to predefined configurations. A major part is the use of a static IP address. Instead of retrieving a network address automatically, a static (fixed) one is used. The domain *scansite4.mit.edu* was also mapped on this one specifically defined address. However, once a *Linux* operating system is installed, the address still has to be configured manually in the OS of the computer. The script setup simply uses one particular address per default.

3.12 *Scansite* Web Service

Next to the *Scansite* main application, also a so called *Web Service* has been available in *Scansite* 3. Due to changes in the main application, the web service was outdated and was not able to work anymore. This additional service is a way to access *Scansite* computationally. By using *Scansite* as a library, a *REST* interface is able to receive requests and execute searches. During the development of *Scansite* to version 4, many functions and interfaces changed. As a consequence the library used by the web service was obsolete. Just replacing the library by a new version did not work due to new interfaces for function calls. Accordingly, web service calls had to be redefined to match current *Scansite* structures. Once available web service calls were redefined, also optional parameters were introduced. This way not every parameter had to be included in the REST service call and could be skipped. In previous versions however, fields had to be set, but their values had to be blank. By using optional parameters, it is now possible to omit parameters, that are not relevant for a call and use default values. Additionally it is possible to offer choices for specific calls. For instance, it is possible to enter either a protein database and an identifier or a protein name together with a sequence as input parameters in the web service link for a protein scan.

The new web service is also the primary test area for *Scansite* in general. Unfortunately, the application did not have any test cases to prove correct functionality in

version 3. Apart from improvements to *Scansite* 4, more than 40 *Unit Test* cases cover tests to ensure functionality and specific outcomes of effectively all features of *Scansite*. This includes different search types, use of default parameters, use of different choices of particular searches and many possible parameter settings.

With the new database access management, the *Scansite Web Service* is now potentially able to do bulk searches. Whereas the previous version restricted searches to approximately 6 jobs before exceeding the number of database connections, it is now possible to computationally access the web service to run hundreds of scans in a row. With the new robust system it is easily possible, to include *Scansite* in computational pipelines, which execute multiple evaluation and analysis steps.

Next to the server side, also the client side part of the web service, which is available on the web (https://scansite4.mit.edu/webservice/), received an upgrade. Apart from applying the new *Scansite* design to the web service, example calls were updated. Upgrading the design included general formatting such as highlighting of examples and demos (see Figure 3.10). On top of that, measures to make understanding web service calls easier were taken. To better comprehend how these calls are structured, easy predefined examples with links are provided. These links as visible in Figure 3.10 call some *JavaScript* code, that runs in the background. This hidden code sends a request with a particular link to the web service server and the results are automatically received, processed and displayed in a result section, that appears after clicking on the link.

Another improvement is a better feature specific description, showing which parameters are supposed to be combined, which ones should not be used together and which ones are highly recommended to use. If endorsed options are not applied, computation time is significantly increased. As the web service in *Scansite* 3 did not offer any possibility to navigate to any other pages including the *Scansite* main page, the navigation bar, that is also available in the main application was added to the web service to be able to go from the core application to the web service and the other way round. Two new features were also added to the web service. This includes the ortholog scan and the prediction of cellular location of a protein. A final point, that improves the web service compared to the one in *Scansite* 3 is encrypted traffic. If a user does not want anyone to know about a certain test protein, a web service call as well as requests in the main application are now both encrypted and in either case input data is discarded after running the search. Additionally, results are also deleted after a certain amount of time.

3.12.1 Web Service Client

Usually just providing a web service is not enough to make people use it. Providing example links to call the REST interface and even *JavaScript* examples is a good step towards usability. However, it does still not show how a program is able to access the service. One of the greatest benefits of this application is certainly the opportunity to run multiple searches at once. However, if one had to copy each and every link into the web browser manually, nobody would ever use this service. This

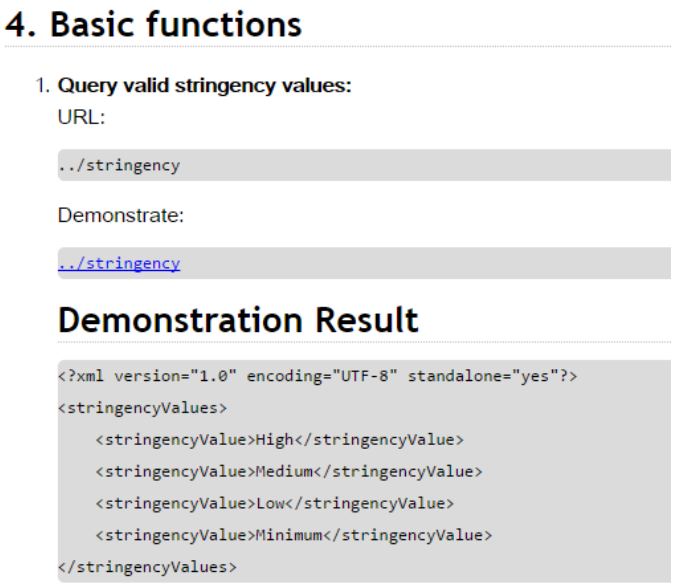


Figure 3.10: The new *Scansite* 4 *Web Service* offers interactive demonstrations. Next to improving the style of the website, also new features are available. By clicking on the blue highlighted link, the *Scansite* REST interface is called with the link *https://scansite4.mit.edu/webservice/stringency* and an appropriate response, which includes the result, is displayed in the *Demonstration Result* section below.

would even be more of an effort than using the *Scansite* core website. Consequently, a *Java* client was implemented. This client is able to execute all example calls, that are listed on the web service web page. Additionally the software client also provides samples for running multiple proteins sequentially by starting the application just once. At this point the *Java* client software is not publicly available, although this might change in future.

3.12.2 Restructuring the *Scansite* Project

At the time, the development of the *Scansite Web Service* started, the core *Scansite* application had to be build, exported as a *Java Archive* (.jar) file and imported as a library to the web service. Based on this setup, a web service development routine had to start each time, changes in the actual *Scansite* application were made. As running this routine could be very exhausting, the *Maven* project structure was extended and partially redefined. Instead of having two completely different and independent projects, which in fact were depending on each other, regarding the library coherence, a common project structure was implemented. On top of the project hierarchy a *Scansite project* was defined, which basically was a collection of *Scansite* related software. After several iterations, the *Scansite project* contained the following four different modules:

- Scansite4Application
- Scansite4DatabaseManager
- Scansite4MotifInserter
- Scansite4WebService

At the beginning only the first two modules were part of this common *Scansite* 4 *project* to be able to deploy the web service without permanently updating the referenced library. The new structure allowed the service to access and build the code of the actual *Scansite* application directly by defining it as a *Maven* dependency. Consequently any changes were automatically inherited by the web service and also adjusting both modules to each other became easier as one could immediately see, when changes in the *Scansite* application module broke features in the web service.

As restructuring the entire project turned out to be highly beneficial, the database manager module was added next. Previously it was necessary to export the *Scansite* application as JAR file, which was only possible due to functions, that were specifically provided by IDEs. Unfortunately, generating a file to run the database setup routine was not able any other way.
Besides, it was exhausting to generate the library for the web service, each time, it was about to be edited or built. Usually the *Scansite* core application was changed in the meantime, which resulted in a deprecated library. To be able to use the latest library, an import and update was essential each time, the web service was further developed. Subsequently, the project was extended by another module. According to the motto *never touch a running system*, the database setup classes have not been separated from the *Scansite* application. The *RunDatabaseInsertionManager* however, could easily be moved over to another module, as no other class had dependencies on it. Similar to the *Scansite4WebService* module, the actual application was set as a *Maven* dependency, to be able to use any functionality provided by *Scansite*. An extra *Maven* module offers the possibility to generate another archive file (WAR or JAR). Thus a separate JAR file for setting up the database with correct configurations (e.g. main program to execute) is now generated each time, the project is built.

As a *nice to have* benefit, the motif inserter was also implemented to be able to process motif PSSMs separately from the main application. Similar to the process of the database manager, a class that calls the actual motif inserter was written and the target class, that does all the work, was called from the *Scansite* application module. Again, this module was used as *Maven* dependency to be able to freely access code from the core program.

Chapter 4

Results and Discussion

This chapter is more user oriented than the previous ones. Whereas the Background chapter gives technical and biological insights, and the Implementations and innovations part focuses on a technical implementation at an abstract level, the Results concentrate on the outcome, that is available to (future) *Scansite* users. This includes new options in input masks, new elements in result pages, visual enhancements, secure traffic and also a more attractive web service. Also very outstanding changes, that are relevant to future developers or researchers with private access are stressed in this part. With respect to user oriented aspects, improvements on the main *Scansite* application, as described in the following sections, certainly have the biggest impact.

4.1 *Scansite* 4

Due to several enhancements and innovations, which were described in the previous chapter, *Scansite* 4 was able to improve outstandingly. Although important points of the upgrade have been explained on a rather technical side, the actual outcome in the user perspective has to be considered as well. As major results of this work focus on advancing the application itself, comparing *Scansite* versions is most likely a sensible way to demonstrate progress. Enhancements are spread throughout the entire application.

A minor example is a rework of the navigation bar. Whereas *Scansite* 3 had a single block to navigate through, the new application has split navigation related contents between navigation bar itself and a footer as depicted in Figure 4.1. The new *Scansite* 4 navigation bar focuses on actively using the application. Hereby effectively any user is welcome to make use of the entire navigation menu. Next to two newly introduced features (*Scan Orthologs* and *Predict Localization*), also the tutorial pages have been split into multiple smaller sub pages to give a better overview for specific functions of the *Scansite* application (see 4.1b). Updated content for tutorials it not available yet, but is planned to be introduced in near future. Navigation content, that is still important, but not related to directly making use of the latest main application are now located in the footer of the website. Contents in the footer can be split into three parts: (i) Retrieving additional information, (ii) accessing the *Administrator and Collaborator Area*, and (iii) going to other *Scansite* related websites such as the

web service or previous versions. Further information can be obtained by visiting the *About*, *FAQ* or *Citing Scansite* sub pages. Particularly general information about the application, credits, publications and support for certain predefined structures like structuring user defined motifs regular expressions are provided on those pages. The second substantial element is the previously mentioned *Administrator and Collaborator Area*. This link loads a login page. If a person has an account, additional content is accessible. Such content includes access to motif management, use of private motifs throughout the application and in case, administrative privileges are granted, also user management is available after logging in. Last but not least links to other *Scansite* related applications are provided in the footer section. The probably most relevant link is the one, that references the new *Scansite* 4 *Web Service*, which is described in the Implementations and innovations chapter. In addition the relevance and meaning of the web service upgrade are underlined later in this chapter (see 4.2). Other links in the footer of the website to previous versions of *Scansite*. Referenced versions include *Scansite* 2 and *Scansite* 3 (see Figure 4.1c).

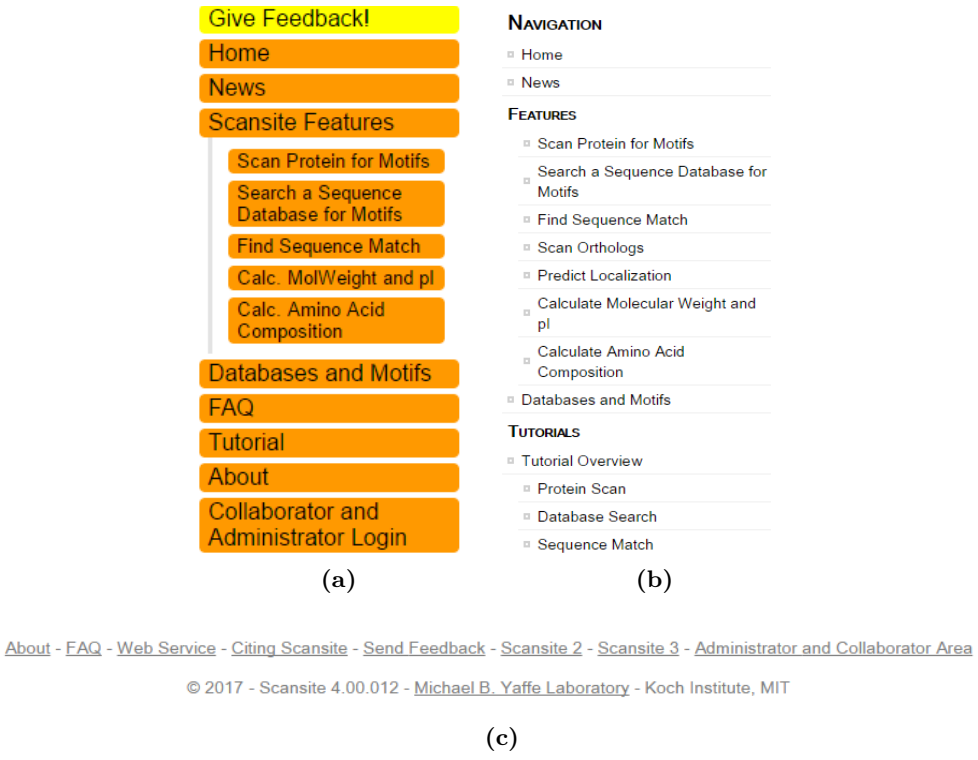


Figure 4.1: Whereas *Scansite* 3 had one central navigation block (Figure 4.1a), *Scansite* 4 introduced additional sub pages and split the navigation into navigation bar and footer at the bottom of the page. The new navigation bar includes elements, that are supposed to be used most often by future users (Figure 4.1b). Additional links to further information or other *Scansite* related applications are available in the *Scansite* 4 footer (Figure 4.1c).

4.1.1 Latest Databases

Another good way to compare both *Scansite* versions is the age of their databases respectively. Whereas the *Scansite* 3 database server has not been updated within the last five years, *Scansite* 4 uses latest releases of mirrored databases (May 2017) and is able to easily update them any time during system maintenance. To give a better impression of the differences within the databases, Figure 4.2 compares *Scansite* 3 to its enhanced version.

Databases

Name	Size (# Proteins)	Version / Last Update
Ensembl Human	92011	2011/11/25
Ensembl Mouse	55797	2011/11/25
NCBI Protein - GenPept/RefSeq	8797684	2011/11/25
SwissProt	533049	UniProt Release 2011_11
Trembl	18215214	UniProt Release 2011_11
Yeast - SGD	6716	2011/11/25

(a)

Databases

Name	Data source type	Size (# of proteins)	Version description	Identifier type
Ensembl Human	Protein	97919	2017/5/19	Ensembl Protein ID
Ensembl Mouse	Protein	65888	2017/5/19	Ensembl Protein ID
LocTree3 (ROSTLAB)	Localization	1851201	2017/5/19	UniProt Entry Name
NCBI HomoloGene	Orthology	257107	Release 68	GenBank Protein Accession
SwissProt Orthology	Orthology	543638	UniProt Release 2017_05	UniProt Entry Name
UniProtKB/Swiss-Prot	Protein	410327	UniProt Release 2017_05	UniProt Entry Name
Yeast - SGD 3	Protein	6680	2017/5/19	SGD Systematic Name

(b)

Figure 4.2: The difference in database age and composition between *Scansite* 3 and *Scansite* 4 is stressed. Whereas *Scansite* 3 is limited to protein databases, *Scansite* 4 also offers orthology and localization databases. At this point the two databases *NCBI Protein - GenPept/RefSeq* and *Trembl* are not available, due to issues with long time downloads. Nevertheless, it is planned to re-enable those two databases in future. More importantly, *Scansite* 3 databases are outdated. Whereas *Scansite* 4 databases have been recently updated in May (2017/5/19), the last update of *Scansite* 3 war in November 2011 (2011/11/25). Obviously sizes of all databases changed outstandingly.

4.1.2 Scan Protein for Motifs

Major improvements were applied to all *Scansite* core features, including the protein scan. Next to visual enhancements, particular elements of the input mask changed remarkably. This includes the options Show predicted domains and Show only previously mapped sites. Both fields can be selected or omitted for a search depending on the intent of the user. Although having additional information using the protein domain scan is certainly a nice thing to have, it also consumes quite some time (approximately 30 seconds–1 minute). The filter option for phosphorylation sites does not need a remarkable amount of additional time, but reduces the output

list by filtering out all results, that do not show an evidence data entry. If just results with primarily great scores are relevant, one might not want to apply a filter. Nevertheless, both options come with new opportunities and also have their advantages coming with implemented innovations. Those particular benefits for a user are pointed out in the sections below.

InterProScan

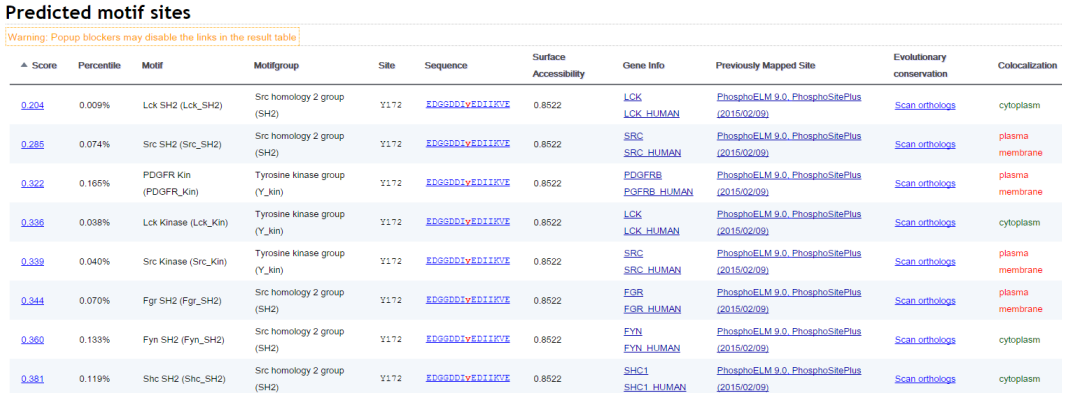
Protein domain scans use more recent data, since *InterPro* databases were updated together with the *InterProScan* application. Hence protein domain prediction is more up to date and probably also more reliable. As the protein databases have also been updated, latest protein sequences can be used as input for the domain scan. This benefit makes results of the *InterPro* third party application even more reliable. The second big advancement regarding protein domains is the newly introduced table, that is able to provide additional information. Especially *IPR Codes*, that are able to provide additional data in new browser tabs, are very useful and reliable values. As protein domains sometimes have big distances to each other within the sequence, having an overview in a table can also be a huge benefit. After all, the greatest advantage of the extended domain search is information about protein families, which is indirectly available via *IPR Codes*.

Phosphorylation Filter

This feature is able to prove, that also other sources have confirmed specific phosphorylation sites. Instead of scrolling through a very long list of results or instead of downloading and processing results, that might not even show up due to stringency restrictions, providing a filter is way easier. Moreover, applying this feature does not cause any delay and results are displayed immediately after running the protein scan. As previously mentioned, this filter opportunity is particularly handy, if connections between existing phosphorylation sites are examined. Another good reason for running the filter is checking the reliability of *Scansite* results in general. Getting an overview of all previously mapped sites for a particular protein can be achieved easily by applying this very filter option. This was done in Figure 4.3 for demonstration purposes.

4.1.3 Search a Sequence Database for Motifs

The database search is the feature, that received the most visual innovations. One of the most obvious changes is about the *Protein Annotations* column of the results table. As one can see by comparing Figure 4.4a and Figure 4.4b, descriptions have received a rework. Instead of writing out the entire text to a single field without particular formatting as implemented in *Scansite* 3, the new appearance focuses on giving a better overview. Hereby only the first approximately 50 characters are displayed by default and it is possible to display a complete and well formatted description by clicking in the cell (e.g. on the *click to expand* text). After clicking, an information box appears (see Figure 4.4c), which contains all protein relevant information. On top any data is divided into separate lines by categories.



Predicted motif sites

Warning: Popup blockers may disable the links in the result table

Score	Percentile	Motif	Motifgroup	Site	Sequence	Surface Accessibility	Gene Info	Previously Mapped Site	Evolutionary conservation	Colocalization
0.204	0.009%	Lck SH2 (Lck_SH2)	Src homology 2 group (SH2)	Y172	EDGSDDIyEDIIKVE	0.8522	LCK LCK_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	cytoplasm
0.285	0.074%	Src SH2 (Src_SH2)	Src homology 2 group (SH2)	Y172	EDGSDDIyEDIIKVE	0.8522	SRC SRC_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	plasma membrane
0.322	0.165%	PDGFR Kin (PDGFR_Kin)	Tyrosine kinase group (Y_kin)	Y172	EDGSDDIyEDIIKVE	0.8522	PDGFRB PGFRB_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	plasma membrane
0.336	0.038%	Lck Kinase (Lck_Kin)	Tyrosine kinase group (Y_kin)	Y172	EDGSDDIyEDIIKVE	0.8522	LCK LCK_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	cytoplasm
0.339	0.040%	Src Kinase (Src_Kin)	Tyrosine kinase group (Y_kin)	Y172	EDGSDDIyEDIIKVE	0.8522	SRC SRC_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	plasma membrane
0.344	0.070%	Fgr SH2 (Fgr_SH2)	Src homology 2 group (SH2)	Y172	EDGSDDIyEDIIKVE	0.8522	FGR FGR_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	plasma membrane
0.360	0.133%	Fyn SH2 (Fyn_SH2)	Src homology 2 group (SH2)	Y172	EDGSDDIyEDIIKVE	0.8522	FYN FYN_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	cytoplasm
0.381	0.119%	Shc SH2 (Shc_SH2)	Src homology 2 group (SH2)	Y172	EDGSDDIyEDIIKVE	0.8522	SHC1 SHC1_HUMAN	PhosphoELM 9.0, PhosphoSitePlus (2015/02/09)	Scan orthologs	cytoplasm

Figure 4.3: A protein scan was run for the protein *VAV2_HUMAN* with mostly default settings. The only additional parameter setting, that differs from its default is the phosphorylation filter. To be able to demonstrate the filter, results are listed in the protein scan result table. This way connections between specific motifs or also between phosphorylation sites within a particular protein can be found out.

Next to the description there are some more relevant changes. For instance the molecular weight did not bear a relation to any unit. As it was not explicitly described as Dalton (Da), the content of the column was not self-explanatory. Consequently, kilo Dalton (kDa) was introduced as a unit and the huge numbers were divided by 1000 accordingly. At this point two decimal places are available. A third and last enhancement compared to *Scansite* 3 is the re-formating of the central phosphorylation site position. To stress this residue even more, *Scansite* underlines it by formatting it as bold red letter.

As depicted in Figures 4.4b and 4.4c, a *Previously Mapped Sites* column was also introduced to the *Scansite* 4 database search. Previously, this feature had only been available in the protein scan. It is most likely as important for the database search as for the protein scan to underline *Scansite* results with existing phosphorylation site evidence. As in the protein scan, links to other websites with detailed information are provided for all available entries in the column. Since each evidence detail page is opened in a new web browser tab, it is possible to check multiple evidence pages without having to use the browser navigation.

Phosphorylation Filter

After introducing the new column to the database search, also the filter for *Previously Mapped Sites* in the input mask was added to this particular *Scansite* feature. Implementing a filter for the database search is probably even more relevant as it is for the protein scan. By adding it to the database search, results for a single motif are displayed and all of them are underlined with an existing phosphorylation site. By applying the filter, connections or similarities between different proteins or previously examined sites can be discovered, which might lead to new models and hypotheses.

Scan this Protein!	▲ Score	Accession	Protein Annotations	Site [14-3-3 Mode 1]	Sequence [14-3-3 Mode 1]	Molecular Weight	pI
Scan!	0.075	PI3R5_HUMAN	Description: RecName: Full=Phosphoinositide 3-kinase regulatory subunit 5; Short=PI3-kinase regulatory subunit 5; AltName: Full=PI3-kinase p101 subunit; AltName: Full=Phosphatidylinositol-4,5-bisphosphate 3-kinase regulatory subunit; Short=PtdIns-3-kinase regulatory subunit; AltName: Full=Protein FOAP-2; AltName: Full=PtdIns-3-kinase p101; AltName: Full=p101-PI3K;; Keywords: Acetylation, Phosphoprotein, Cytoplasm, Polymorphism, Membrane, Reference proteome, Complete proteome, Nucleus, Alternative splicing; Accessions: Q9Y2Y2, Q8WYR1, Q5G938, Q5G936, B0LPH4, D3DTS3, Q5G939, Q8IZ23;	S478	SRAQRSRsLPQPKLG	97360.2	6.30
Scan!	0.079	CC125_HUMAN	Description: RecName: Full=Coiled-coil domain-containing protein 125; AltName: Full=Protein kenae;; Keywords: Cytoplasm, Polymorphism, Coiled coil, Reference proteome, Complete proteome, Alternative splicing; Accessions: Q86Z19, Q86ZZ0;	S504	RTLKRSHsLPSSIIF	58635.8	6.86
Scan!	0.083	CCR6_HUMAN	Description: RecName: Full=C-C chemokine receptor type 6; Short=C-C CKR-6; Short=CC-CKR-6; Short=CCR-6; AltName: Full=Chemokine receptor-like 3; Short=CKR-L3; AltName: Full=DRY6; AltName: Full=G-protein coupled receptor 29; AltName: Full=GPR-CY4; Short=GPRCY4; AltName: Full=LARC receptor; AltName: CD_antigen=CD196;; Keywords: Transducer, Glycoprotein, Cell membrane, Disulfide bond, Transmembrane helix, G-protein coupled receptor, Membrane, Reference proteome, Receptor, Transmembrane, Complete proteome; Accessions: P51684, Q92846, E1P5C6, P78553;	T160	SFRLRSRtLPRSKII	42499.5	9.23
Scan!	0.083	NHS_HUMAN	Description: RecName: Full=Nance-Horan syndrome protein; AltName: Full=Congenital cataracts and dental anomalies protein;; Keywords: Phosphoprotein, Cataract, Polymorphism, Reference proteome, Complete proteome, Nucleus, Alternative splicing; Accessions: Q6BDR5, Q5J7Q1, Q5J7Q0, Q6T4R5;	S485	SSRTRSRsLPREGNR	176723.5	6.32
Scan!	0.091	NHSL1_HUMAN	Description: RecName: Full=NHS-like protein 1;; Keywords: Phosphoprotein, Polymorphism, Reference proteome, Complete proteome, Alternative splicing; Accessions: Q9P2J0, Q5SYE8, Q3ZCS5, Q5SYE7;	S1458	PRFQRSRsEPSPDAP	170691.2	6.52

(a)

Scan this Protein!	▲ Score	Accession	Protein Annotations	Site [14-3-3 Mode 1]	Sequence [14-3-3 Mode 1]	Previously Mapped Site	Molecular Weight [kDa]	pI
Scan!	0.075	PI3R5_HUMAN	RecName: Full=Phosphoinositide 3-kina (click to expand)	S478	SRAQRSRsLPQPKLG		97.36	6.30
Scan!	0.079	CC125_HUMAN	RecName: Full=Coiled-coil domain-cont (click to expand)	S504	RTLKRSHsLPSSIIF	PhosphoSitePlus (2015/02/09)	58.64	6.86
Scan!	0.083	CCR6_HUMAN	RecName: Full=C-C chemokine receptor (click to expand)	T160	SFRLRSRtLPRSKII		42.50	9.23
Scan!	0.083	NHS_HUMAN	RecName: Full=Nance-Horan syndrome pr (click to expand)	S506	SSRTRSRsLPREGNR	PhosphoSitePlus (2015/02/09)	179.16	6.40
Scan!	0.091	NHSL1_HUMAN	RecName: Full=NHS-like protein 1;; Ac (click to expand)	S1458	PRFQRSRsEPSPDAP	PhosphoSitePlus (2015/02/09)	170.69	6.52

(b)

Scan this Protein!	▲ Score	Accession	Protein Annotations	Site [14-3-3 Mode 1]	Sequence [14-3-3 Mode 1]	Previously Mapped Site	Molecular Weight [kDa]	pI
Scan!	0.075	PI3R5_HUMAN	RecName: Full=Phosphoinositide 3-kina (click to expand)	S478	SRAQRSRsLPQPKLG		97.36	6.30
Scan!	0.079	CC125_HUMAN	RecName: Full=Coiled-coil domain-cont (click to expand)	S504	RTLKRSHsLPSSIIF	PhosphoSitePlus (2015/02/09)	58.64	6.86
					SFRLRSRtLPRSKII		42.50	9.23
					SSRTRSRsLPREGNR	PhosphoSitePlus (2015/02/09)	179.16	6.40
Scan!	0.091	NHSL1_HUMAN	RecName: Full=NHS-like protein 1;; Ac (click to expand)	S1458	PRFQRSRsEPSPDAP	PhosphoSitePlus (2015/02/09)	170.69	6.52

Description:
RecName: Full=Coiled-coil domain-containing protein 125
AltName: Full=Protein kenae
Accessions: Q86Z19, Q86ZZ0
Keywords: Alternative splicing, Complete proteome, Reference proteome, Phosphoprotein, Cytoplasm, Coiled coil, Polymorphism

(c)

Figure 4.4: The result page of the database search in *Scansite* 3 as depicted in Figure 4.4a was remarkably changed. *Scansite* 4 introduced a unit for the molecular weight, which uses another scaling (see Figure 4.4b). Moreover the zero position of each phosphorylation site match was stressed by using a bold red letter. Another important change is the description, which was a single unformatted block in *Scansite* 3. The new program displays only around 40 characters (see Figure 4.4b) and is able to be expanded. By clicking on the description cells *Scansite* 4 displays an additional rectangle, which shows the complete protein annotation well formatted (see Figure 4.4c). Each annotation category is started in a new line. Last but not least Figures 4.4b and 4.4c depict the newly introduced *Previously Mapped Sites* column, which had already been available in the protein scan in *Scansite* 3.

Modification Dependent Phosphorylation Sites

Together with the new motif structure, which was introduced during the development of *Scansite* 4, modifications on non-zero motif positions were introduced. Next to the possible use of these values also a number of private motifs was added to the *Scansite* motif collection. Those new motifs have additional columns for modified residues. This includes phosphorylated residues such as pS (phosphorylated Serine), pT (phosphorylated Threonine) and pY (phosphorylated Tyrosine). These innovations allow additional matches for new motifs, since modified residues are scored differently, as depicted in Figure 4.5. Whereas experience is required for a protein scan to choose a protein, that actually matches one of the new motifs, it is easier to find a new site in the database search. If one of the new motifs is selected as input for

the search, probably necessary modifications before or after the core position in the 15 residue window are highlighted in orange color. These additional modifications might be conditional for the actually focused phosphorylation site to be part of a reaction. Additional modifications can have influence on molecular forces or accessibility within the 3D (three dimensional) protein structure. Subsequently, a missing relevant conditional modification might prevent phosphorylation from happening.

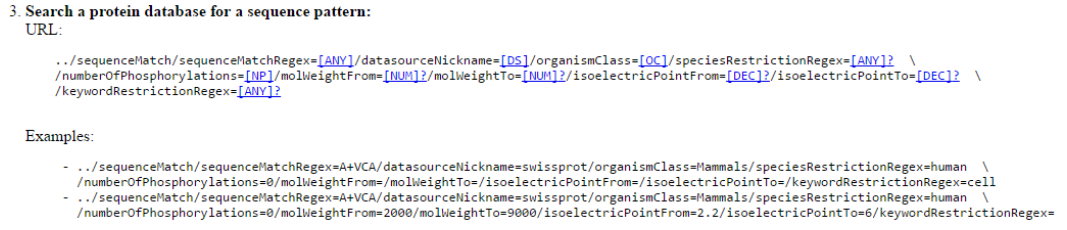
0.351	HNRPK_HUMAN	RecName: Full=Heterogeneous nuclear r (click to expand)	S372	GGSGyDysyAGGRGS	PhosphoSitePlus (2015/02/09)
0.353	DPP4_HUMAN	RecName: Full=Dipeptidyl peptidase 4; (click to expand)	S333	DICDYDEsSGRWNCL	
0.357	OTOF_HUMAN	RecName: Full=Otoferlin; AltName: Ful (click to expand)	T47	DVADFDEtFRWPVAS	
0.370	TUT7_HUMAN	RecName: Full=Terminal uridylyltransf (click to expand)	T882	NEDELDNtYTGSGDE	
0.371	ZZZ3_HUMAN	RecName: Full=ZZ-type zinc finger-con (click to expand)	S22	GLNGLDEsFCGRTLR	
0.380	CBPB1_HUMAN	RecName: Full=Carboxypeptidase B; EC= (click to expand)	T271	SRNPCDEtYCGPAAE	
0.383	TM131_HUMAN	RecName: Full=Transmembrane protein 1 (click to expand)	T1855	PADDLGQtyNPWRIW	
0.389	JKIP2_HUMAN	RecName: Full=Janus kinase and microt (click to expand)	S468	PDDDLDEsLAAEESE	
0.395	ST65G_HUMAN	RecName: Full=STAGA complex 65 subuni (click to expand)	S387	SPDDSDSsYGSHSTD	
0.398	SYNE1_HUMAN	RecName: Full=Nesprin-1; AltName: Ful (click to expand)	S8369	tGPELDtsYKGyMKL	
0.398	SYNE1_HUMAN	RecName: Full=Nesprin-1; AltName: Ful (click to expand)	S8369	tGPELDtsYKGyMKL	
0.399	ABLM3_HUMAN	RecName: Full=Actin-binding LIM prote (click to expand)	S514	EEDDFDRsMHKLQSG	PhosphoSitePlus (2015/02/09)

Figure 4.5: An extract of a database search based on a new private motif is depicted. Hereby *Nek6* was used as a motif so search for. Since new motifs have additional values for phosphorylated residues, potential modifications before and after the phosphorylation site of interest show up. These are marked in orange lower case letters and indicate phosphorylated residues, that might be considered conditional modifications. An additional phospho group is able to change molecular forces and thus the protein structure. As a consequence it could be possible, that the focused site may or may not be accessible depending on surrounding modifications.

4.2 *Scansite* 4 Web Service

Apart from the *Scansite* main application also the web service plays an important role. After updating functionality, this additional program offers significantly more than its predecessor. First of all, navigation elements to go back to the main program are now available. This makes it easy to test some parameter settings manually before accessing the web service computationally. Highlighting relevant elements with style settings is also an improvement, which helps to get a better overview of the page. Moreover *JavaScript* demonstrations make the use of the web service much easier. Predefined examples are textually provided and just by clicking on a link, a request is sent to the server and results are automatically displayed. As this program is designed to access *Scansite* computationally as well as to do bulk searches, an example client is also available. This web service client is at least accessible to all members with a *Scansite* account or on request. With additional search options (*Scan Orthologs* and *Predict Localization*), such as optional parameters and a general rework in the background, the *Scansite Web Service* has evolved to a new level of usability and makes a good baseline for future development.

As described in previous chapters, also several visual enhancements were introduced. Figure 4.6 compares the previously used web service page (see Figure 4.6a) to its new *Scansite* 4 version (see Figure 4.6b). It shows how especially the general structure of a web service request and example calls can be stressed to give a better overview. By using a slide bar instead of *Linux* based syntax to indicate, that the command still continues in the following line, the structure of the website seems to be less complex and commands can be understood more easily.


3. **Search a protein database for a sequence pattern:**
URL:

```
../sequenceMatch/sequenceMatchRegex=[ANY]/datasourceNickname=[DS]/organismClass=[OC]/speciesRestrictionRegex=[ANY]?  \
/numberOfPhosphorylations=[NP]/molWeightFrom=[NUM]?/molWeightTo=[NUM]?/isoelectricPointFrom=[DEC]?/isoelectricPointTo=[DEC]?  \
/keywordRestrictionRegex=[ANY]?
```

Examples:

```
- ../sequenceMatch/sequenceMatchRegex=A+VCA/datasourceNickname=swissprot/organismClass=Mammals/speciesRestrictionRegex=human  \
  /numberOfPhosphorylations=0/molWeightFrom=/molWeightTo=/isoelectricPointFrom=/isoelectricPointTo=/keywordRestrictionRegex=cell
- ../sequenceMatch/sequenceMatchRegex=A+VCA/datasourceNickname=swissprot/organismClass=Mammals/speciesRestrictionRegex=human  \
  /numberOfPhosphorylations=0/molWeightFrom=2000/molWeightTo=9000/isoelectricPointFrom=2.2/isoelectricPointTo=6/keywordRestrictionRegex=
```

(a)


3. **Search a sequence database for a motif:**
URL:

```
../databasesearch/motifshortname=[M]/dsshortname=[DS]/organismclass=[OC]/speciesrestriction=[ANY]?/numberofphosphorylations=[NP]/molweightfrom=[NUM]?/molweight
```

Optional parameters:
- speciesrestriction
- numberofphosphorylations
- molweightfrom
- molweightto
- isoelectricpointfrom
- isoelectricpointto
- keywordrestriction
- sequencerestriction

Please use at least the species restriction to reduce runtime!
Examples :

```
- ../databasesearch/motifshortname=Shc_SH2/dsshortname=swissprot/organismclass=Mammals/speciesrestriction=rattus/numberofphosphorylations=1/molweightfrom=5000
- ../databasesearch/motifshortname=Shc_SH2/dsshortname=swissprot/organismclass=Mammals/numberofphosphorylations=0
```

(b)

Figure 4.6: An extract of the web service *Scansite functions* is shown. Hereby the database search was selected. Image 4.6b makes clear what mandatory and what optional parameters can be used. The general structure of the call is shown in the first gray highlighted box whereas applicable examples are listed further below. For scaling reasons, the image shows only parts of the complete request strings. Due to new formatting elements appear to be more intuitive and are supposed to be understood more easily. This can be particularly stressed by comparing Figures 4.6a and 4.6b. Especially when it comes to the general call structure and example calls, backslashes as used in *Linux* terminals to indicate, that a command continues on a new line, are not required any longer.

4.3 Database Setup Mechanism

Although several advantages of the database setup rework have been pointed out in previous chapters, it is considered relevant enough to bring up some more points. It is probably one of the most important steps towards future development, as working on a separate database can make introducing new features much easier. Especially future changes in the database or *try and error* tests can be supported with this

routine. Based on configuration files, an anytime database setup can be customized to particular scenarios without downloading and preparing all databases at once. As the new setup deletes any files, that are downloaded after finishing its execution, it can also serve as update mechanism. Even though it takes a couple of hours to finish, the setup allows to maintain the system. All databases could be updated once a year, for instance during new years eve. As long as http and ftp download sources do not switch to https and as long as web addresses for downloads stay the same, the setup can be run any time. Configuring so called *cron jobs* would also be an option to keep databases up to date. A cron job is a process on *Linux* systems, that is automatically and periodically executed (e.g. once a year).

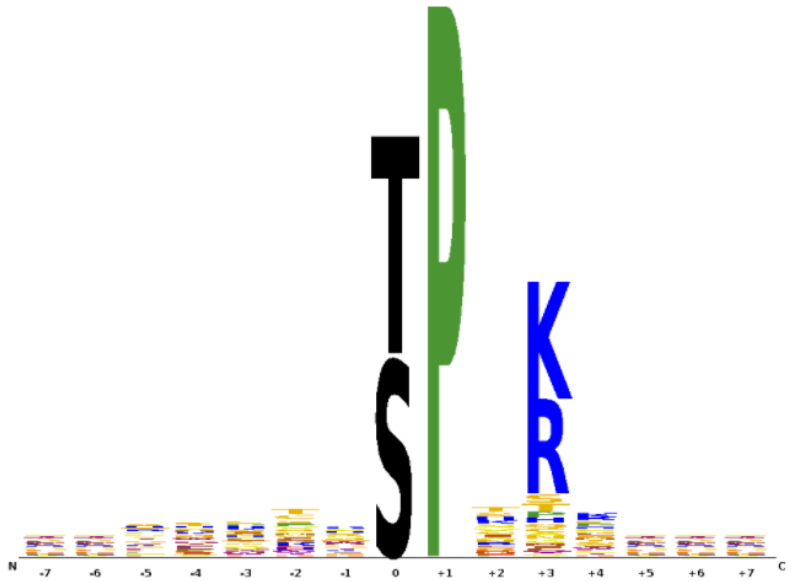
4.4 Database Access Management

Another big achievement is the database access management replacement. It allows more extensive testing of *Scansite* and executing *Scansite* based applications. Moreover running the web service with multiple requests in mind is now also easily possible. The new abstraction layer, that provides a shared connection pool instead of establishing a new one with each *Scansite* feature is probably the most effective and most essential change in this rework. As the limit of database connections is not exceeded anymore, and *Scansite* fully manages database connections itself, other applications do not have to deal with connection management anymore. Every single shared connection is able to be used and reused as often as required. The default number for the size of the connection pool is centrally defined (default 25) and can be changed by editing a single line of code. Additionally, the size of the connection pool can also be set from outside using a function call. For instance while setting up the database, the size of the pool is set to one, as a single long time connection is required due to enabling and disabling of constraints.

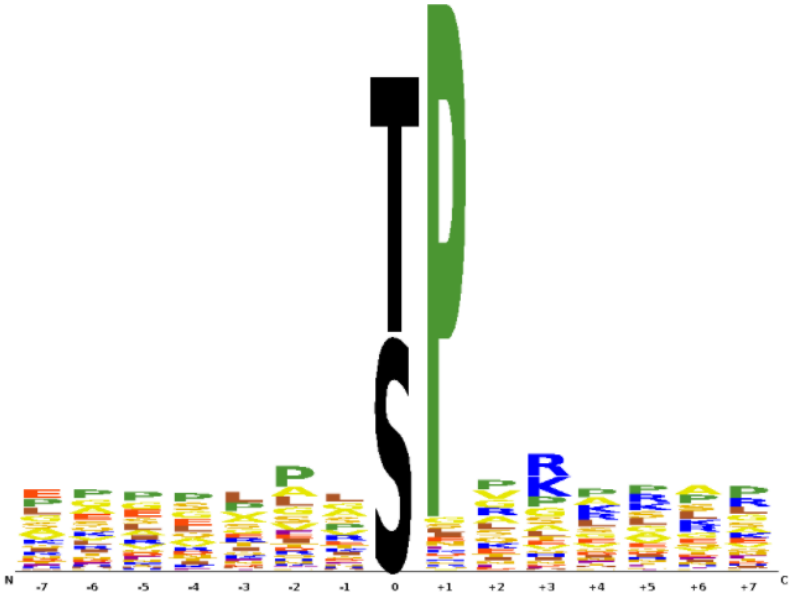
4.5 *In Silico* Motifs

Generated motifs are not necessarily a direct improvement of *Scansite* at the moment. However, methods like this particular one can have significant impact in future. Up to now, OPLS experiments are limited in their positions. This means, that experimental motifs can only be determined for positions from -5 to +4 and further numbers towards N- and C- terminus of the motif rely on experiences of the past. With a larger number of phosphorylation substrates, it would be able to find out general preferences for these positions as well as certain relations, which however could have a great impact on future *Scansite* scores. *In silico* motifs are still a good first step to have additional sources, that stress the correctness of a certain kind of wet lab experiments. To do so, Figure 4.7 compares an experimental version of the *CDK1* motif (see 4.7a) to its generated counterpart (see Figure 4.7b).

Unfortunately generated motifs are not equally reliable. Some phosphorylation substrate pages on the *PhosphoSitePlus* web site offered hundreds of substrate sequences, whereas others had a relatively low number, which does not really provide reliability, nor can it prove any statistical validity.



(a)



(b)

Figure 4.7: Different motifs for *CDK1* are compared. The first one in Figure 4.7a is the publicly available "*CDK1 motif 1 - [ST]Px[KR]x*" motif and the second one in Figure 4.7b is a generated *in silico* motif. Similarities such as the dominant Proline (P) on the +1 position, or Arginine (R) and Lysine (K) on the +3 position, underline the correctness of the peptide library search experiments.

Chapter 5

Possible Further Extensions and Outlook

Scansite 4 as well as all related applications offer – as software – always room for improvement. However, an important and a very influential step has been made by evolving *Scansite* from version 3 to 4. With introducing new features, also potential sources for new issues have to be considered. Although the system runs more reliable, than before, additional complexity has always its up and down sides. Whereas a *Scansite* user has a probably much better experience while actively making use of the application, a new developer probably needs more time to learn the ropes for future enhancements.

As one can not generalize pros and cons for the entire *Scansite* project, certain parts need to be particularly focused. With these functional mechanisms in mind, this discussion primarily considers future development opportunities. As many different parts of the *Scansite* project have changed, primarily *Scansite* application and web service modifications are relevant for discussing advantages, disadvantages and future opportunities. The first important point is about the more or less integrated modifications around the central phosphorylation site.

5.1 Additional Modifications

At this point phosphorylation is fully integrated as *UniProtKB* sequences have been modified and new motifs with separate values for phosphorylated residues are also available, given that a user is logged in. Unfortunately, evidence sources only include phosphorylation at the moment. In future it would be necessary, to find appropriate sources to adjust sequences also to other previously mapped modifications. Another issue is, that only *UniProtKB* is useful for these special searches. All other databases have different identifiers compared to the evidence data. Even more sources or an identifier mapping would be required to achieve sufficient results. Nevertheless, there is a difficult trade off between considering the current state as sufficient and extending the features in a way, that would cover nearly all available protein sequences. Big investments with respect to time and financial support would be necessary to make such an extension possible.

To reach this goal, also additional motifs would be required. If not a single motif offers specific values for modified residues, regardless of indicated modifications in the database, not a single match can be found. To be able to provide such motifs, currently used peptide libraries might have to be replaced, as those being used at the moment do not include modified Arginine or Lysine.

5.2 Evidence Data Update

Unlike databases, evidence data from *PhosphoSitePlus*, *Phospho ELM* and *Phosida* has not been updated in years. Unfortunately, the latter two sources have not been maintained in the meantime and information might be outdated. *PhosphoSitePlus* however, does maintain the related database. Thus it would be possible to update evidence data in future. This would be highly recommended as *Previously Mapped Sites* columns in protein scan and database search as well as any content related to further phosphorylated sites is connected to evidence data.

5.3 Additional Filters

Introducing a filter option in general certainly was a benefit to *Scansite* 4. Displaying only result entries, that have site evidence, stresses the reliability of *Scansite* and shows new connections. As many things, also filter options offer room for improvement. This is referred to the current number of filters. Whereas a single filter is very limited in its options, multiple filters and even combination could lead to even more specific results. Of course, especially in the database search, there are multiple options to restrict the result list. However, one could still introduce additional filters such as show only conditional modified sites or limit the matched results to those, that have proteins with certain accession attributes. For the protein scan the accession filter as well as excluding particular motifs (e.g. logged in and exclude generated motifs) might be useful. Moreover a combination of filters could lead to even more specific results.

Another point about the filters is, that they are applied while searching. It would be more comfortable, if one could just run the search and apply filters to all potential results, including those, that are discarded based on a given stringency value.

5.4 InterProScan

This sub-feature received a remarkable upgrade. *InterPro* databases for protein domain searches were updated, a different version of the third party application *InterProScan* is in use, which has less requirements, and thus is easier to install. On top the new protein domain info table with the *IPR Codes* is available, which enables *Scansite* to link to even further information. Although all these innovations can be described as outstanding progress, connections between elements in the table and domains in the image might still get lost due to overlapping labels in the picture. Hence introducing an additional column with the color codes, that are used in the image would be a useful future implementation.

5.5 Database Setup Mechanism

Providing a setup, that is able to setup the entire *Scansite* database fully automatically by running a single *Java* class makes developing much easier. Also the server setup benefits greatly from this routine, and yet there is still room for improvement. Downloading and processing information from public databases is far from being optimized. The previously applied parallel approach caused issues for the setup in general and ended in crashing the application. Its new version is working reliably as long as databases do not surpass a certain size. After spending approximately half an hour for downloads, loading further data stops and the updater does not continue. It switches to an idle state instead and has to be restarted. Next to this download issue, time management offers room for improvement. A system with two queues would be a suitable solution to the huge amount of time, that is currently required. One of the queues would be a download queue, that is in charge of receiving required data from the web, saving it to a temporary folder and marking downloads as complete. The second queue would process locally available data and inserts it into the local *MySQL* database. If larger processing steps before actually inserting data into the database are needed, even a third queue could be introduced to split the two tasks.

Next to optimizing the process itself, keeping databases up to date is also relevant. As mentioned in the previous chapter, a *cron job* would be a great solution. It would be possible to maintain and update the system periodically and fully automatically, assuming that databases are still available in future and download links and protocols are not changed. By generating such a cron job, it would be possible to update the system (including operating system updates) for instance once in three months or at least once a year.

5.6 Web Service

The *Scansite Web Service* has evolved outstandingly during the development of *Scansite* 4. The frontend (the web page) is much more intuitive and easier to understand, and the backend on the server side is way more reliable. Even though the application is now easier to use, especially for people who are able to access the *Java* client, it is not very useful for a researcher without a programming background. As a consequence, a major improvement is an additional service, that allows similar *Scansite* website searches new search methods. Those should work just like the original *Scansite* feature with one big difference. The new *Scansite* project module should be able to use *Scansite* or even the web service to run bulk searches by for instance simply uploading a file with all proteins of interest. Any other parameters should be able to be adjusted in the web interface. Although such an improvement is not available yet, it is planned to be implemented in near future.

Another issue, that should be tackled is a general approach to deal with web service requests. As long as requests are in a correct format and as long as all parameters are set correctly, everything works perfectly fine. However, incorrect parameter settings like typos result in a 404 HTTP error, which means, that a requested sources could not have been found. The *Scansite* web service is not able to assign incorrect requests

at all. Instead of returning a signal, that no feature could have been assigned, the service should assign requests to a default interface. This default interface should be able to deal with all non-mapped requests. It should further recognize the selected feature to answer more specifically. The response of the default interface should include instructions how a specific call should be put together, depending on the previously recognized feature respectively.

5.7 *In Silico* Motifs

Currently private *in silico* motifs are available for all users with a login. Unfortunately old motifs do not provide information about exact values of their zero position. New motifs, which do provide such information are not suitable for generating *in silico* motifs. At this point, there are no sufficient phosphorylation substrate pages on the *PhosphoSitePlus* website to build appropriate motif PSSMs. However, in future such values might be available or central position values for existing motifs might be determined and added. Although it is not beneficial at this point, adding real values to the central position of *in silico* motifs for the motif logos would be a sensible innovation in future.

Chapter 6

Conclusion

Based on described sections of the Results chapter and also with respect to pros and cons in the Possible further extensions and outlook chapter it can be pointed out, that *Scansite* has improved significantly from version 3 to 4. Important changes made the application more intuitive, the design looks more elegant and gives a better overview and functionality has improved remarkably too. Especially the opportunity to consider modified residues, mostly phosphorylates amino acids at this point, is a huge step towards better predicting correct phosphorylation sites. By having additional potential modifications, it is possible to predict conditions for certain mechanisms to trigger. If this knowledge can be transformed to drug development, *Scansite* can have a big impact in future on patient treatment. Apart from the fundamental change about modified residues, also introducing the *Previously Mapped Sites* column and the phosphorylation residue filter are fragments, which have big impact on search results.

However, there are also smaller changes such as adding a unit to molecular weights, highlighting certain values or giving a better overview in the results table. All those things are nice to have but do not have a big impact for the future of *Scansite*. Splitting the tutorial into sub pages and restructuring content however, is more useful. Elements like the new database access management or the database setup do not have any benefit to the user, but they provide huge comfort for future developers and make things easier and more reliable.

The web service is also a highly relevant part of the application. The new version is more robust, contains tests, which ensure correct functionality of the *Scansite* main application, and is more intuitive due to the visual rework. Together with example calls and the *Java* client, that might be publicly available in future, the web service as it is becomes a much more attractive application to use.

But, as previously pointed out, there are still many fragments in the project, that offer significant room for improvement. Whereas some of the points mentioned are idealistic, others are certainly able to be realized in a decent amount of time. Especially in science, a much more important question is: What is the most useful improvement? This question is often hard to be answered. At the moment extending *Scansite* functionality, including the main application and the web service, to be able to run bulk analysis is considered most relevant. Other outlooks such as editing

all protein sequence databases to indicate potential phosphorylation or other modifications is probably too time consuming, to actually implement such a feature.

To put all in a nutshell, *Scansite* as a project gained a lot of improvements. The project structure itself has been extended, new sub-functions were added to the main application, the web service works reliably and contains more than 40 tests to ensure correct functionality and a developer has an easy database setup for a fresh start. Additionally the entire server setup is documented and runs almost fully automatically. With all those enhancements given, *Scansite* certainly made a big step in the right direction. In addition working on the project is much more comfortable in future. Of course, there are features, that offer room for improvement. However, software can always be further optimized to work with new technologies, new standards or other sources of data. At this point, *Scansite* is a very useful application and the current source code offers a solid base line to continue developing any time.

References

Literature

[1] Jes Alexander et al. “Spatial exclusivity combined with positive and negative selection of phosphorylation motifs is the basis for context-dependent mitotic signaling.” In: *Science signaling* 4 (179 June 2011), ra42 (cit. on p. 8).

[2] Michael Ashburner et al. “Gene Ontology: tool for the unification of biology”. In: *Nature Genetics* 25.1 (May 2000), pp. 25–29. URL: https://doi.org/10.1038%2F75556 (cit. on p. 37).

[3] Michael J Begley et al. “EGF-receptor specificity for phosphotyrosine-primed substrates provides signal integration with Src.” In: *Nature structural & molecular biology* 22 (12 Dec. 2015), pp. 983–990 (cit. on pp. 7, 9).

[4] Andrew D. Birrell and Bruce Jay Nelson. *Remote procedure call*. Tech. rep. 1981 (cit. on p. 21).

[5] The UniProt Consortium. “UniProt: the universal protein knowledgebase”. In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D158–D169. URL: https://doi.org/10.1093%2Fnar%2Fgkw1099 (cit. on p. 29).

[6] David M. Creasy and John S. Cottrell. “Unimod: Protein modifications for mass spectrometry”. In: *PROTEOMICS* 4.6 (June 2004), pp. 1534–1536. URL: https://doi.org/10.1002%2Fpmic.200300744 (cit. on p. 47).

[7] Pau Creixell et al. “Unmasking determinants of specificity in the human kinome.” In: *Cell* 163 (1 Sept. 2015), pp. 187–201 (cit. on pp. 1, 2).

[8] Holger Dinkel et al. “Phospho.ELM: a database of phosphorylation sites-update 2011”. In: *Nucleic Acids Research* 39 (2011), p. D261 (cit. on p. 21).

[9] Tobias Ehrenberger. “Computational Prediction of Kinase-Substrate Interactions with Scansite 3 and the Enrichment of well-known Protein-Protein Interaction Networks with Novel and Relevant Interactors”. MA thesis. University of Applied Sciences Upper Austria, June 2012 (cit. on pp. 8, 11, 15).

[10] Tobias Ehrenberger, Lewis C Cantley, and Michael B Yaffe. “Computational prediction of protein-protein interactions.” In: *Methods in molecular biology (Clifton, N.J.)* 1278 (2015), pp. 57–75 (cit. on pp. 1–3, 25).

[11] Robert D Finn et al. “InterPro in 2017–beyond protein family and domain annotations”. In: *Nucleic acids research* 45.D1 (2017), pp. D190–D199 (cit. on p. 41).

[12] F. Gnad, J. Gunawardena, and M. Mann. "PHOSIDA 2011: the posttranslational modification database". In: *Nucleic Acids Research* 39.Database (Nov. 2010), pp. D253–D260. URL: https://doi.org/10.1093%2Fnar%2Fgkq1159 (cit. on p. 21).

[13] Tatyana Goldberg et al. "LocTree3 prediction of localization." In: *Nucleic acids research* 42 (Web Server issue July 2014), W350–W355 (cit. on p. 17).

[14] Frank R Hampel et al. "Linear models: robust estimation". In: *Robust Statistics: The Approach Based on Influence Functions* (1986), pp. 307–341 (cit. on p. 15).

[15] K. K. Han and A. Martinage. "Post-translational chemical modification(s) of proteins". In: *Int. J. Biochem.* 24.1 (1992), pp. 19–28 (cit. on pp. 1, 6).

[16] Peter V Hornbeck et al. "PhosphoSitePlus, 2014: mutations, PTMs and recalibrations." In: *Nucleic acids research* 43 (Database issue Jan. 2015), pp. D512–D520 (cit. on pp. 13, 21).

[17] Jessica E Hutti et al. "A rapid method for determining protein kinase phosphorylation specificity." In: *Nature methods* 1 (1 Oct. 2004), pp. 27–29 (cit. on pp. 8, 9).

[18] L. M. Iakoucheva. "The importance of intrinsic disorder for protein phosphorylation". In: *Nucleic Acids Research* 32.3 (Feb. 2004), pp. 1037–1049. URL: https://doi.org/10.1093%2Fnar%2Fgkh253 (cit. on p. 29).

[19] P. Jones et al. "InterProScan 5: genome-scale protein function classification". In: *Bioinformatics* 30.9 (Jan. 2014), pp. 1236–1240. URL: https://doi.org/10.1093%2Fbioinformatics%2Fbtu031 (cit. on pp. 27, 41).

[20] Gang Lu et al. "Phosphorylation of ETS1 by Src family kinases prevents its recognition by the COP1 tumor suppressor." In: *Cancer cell* 26 (2 Aug. 2014), pp. 222–234 (cit. on p. 3).

[21] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Apache Maven*. Alpha Press, 2010 (cit. on p. 5).

[22] John C Obenauer, Lewis C Cantley, and Michael B Yaffe. "Scansite 2.0: Proteome-wide prediction of cell signaling interactions using short sequence motifs." In: *Nucleic acids research* 31 (13 July 2003), pp. 3635–3641 (cit. on pp. 2, 3).

[23] Sue Povey et al. "The HUGO Gene Nomenclature Committee (HGNC)". In: *Human Genetics* 109.6 (2001), pp. 678–680. URL: http://dx.doi.org/10.1007/s00439-001-0615-0 (cit. on p. 20).

[24] Sangya Pundir, Maria J. Martin, and Claire O'Donovan. "UniProt Protein Knowledgebase". In: *Protein Bioinformatics*. Springer New York, 2017, pp. 41–55. URL: https://doi.org/10.1007%2F978-1-4939-6783-4_2 (cit. on p. 20).

[25] M. Rebhan et al. "GeneCards: integrating information about genes, proteins and diseases". In: *Trends Genet.* 13.4 (Apr. 1997), p. 163 (cit. on p. 29).

[26] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. URL: https://doi.org/10.1145%2F359340.359342 (cit. on p. 57).

[27] Peter J Rousseeuw and Christophe Croux. “Alternatives to the median absolute deviation”. In: *Journal of the American Statistical association* 88.424 (1993), pp. 1273–1283 (cit. on p. 15).

[28] Z Songyang et al. “Use of an oriented peptide library to determine the optimal substrates of protein kinases.” In: *Current biology : CB* 4 (11 Nov. 1994), pp. 973–982 (cit. on p. 1).

[29] Michael Widenius and Davis Axmark. *Mysql Reference Manual*. Ed. by Paul DuBois. 1st. Sebastopol, CA, USA: O’Reilly & Associates, Inc., 2002 (cit. on p. 5).

[30] M B Yaffe et al. “A motif-based profile scanning approach for genome-wide prediction of signaling pathways.” In: *Nature biotechnology* 19 (4 Apr. 2001), pp. 348–353 (cit. on pp. 2, 11).

[31] Michael B Yaffe. “How do 14-3-3 proteins work?– Gatekeeper phosphorylation and the molecular anvil hypothesis.” In: *FEBS letters* 513 (1 Feb. 2002), pp. 53–57 (cit. on p. 10).

[32] Y Yarden and M X Sliwkowski. “Untangling the ErbB signalling network.” In: *Nature reviews. Molecular cell biology* 2 (2 Feb. 2001), pp. 127–137 (cit. on p. 7).

Online sources

[33] Feb. 2017. URL: http://www.gwtproject.org/ (cit. on p. 5).