Recognition of Short Functional Motifs in Protein Sequences

Dissertation der Fakultätfür Biologie
der Ludwig-Maximilians-Universität München

Roman Prytuliak
Martinsried, 2017

Diese Dissertation wurde angefertigt
unter der Leitung von Prof. Dr. Christian Leibold
im Bereich von Neurowissenschaft
an der Ludwig-Maximilians-Universität München

Erstgutachter:	Prof. Dr. Christian Leibold
Zweitgutachter:	Dr. Bianca Habermann
Tag der Abgabe:	24.10.2017
Tag der mündlichen Prüfung:	22.06.2018

ERKLÄRUNG

Ich versichere hiermit an Eides statt, dass meine Dissertation selbständig und ohne unerlaubte Hilfsmittel angefertigt worden ist.
Die vorliegende Dissertation wurde weder ganz, noch teilweise bei einer anderen Prüfungskommission vorgelegt.
Ich habe noch zu keinem früheren Zeitpunkt versucht, eine Dissertation einzureichen oder an einer Doktorprüfung teilzunehmen.

Martinsried, den 30.06.2018

Roman Prytuliak

CONTENT

1. INTRODUCTION

1.1. Aims of this study

The main goal of this study was to develop a method for computational *de novo* prediction of short linear motifs (SLiMs) in protein sequences that would provide advantages over existing solutions for the users. The users are typically biological laboratory researchers, who want to elucidate the function of a protein that is possibly mediated by a short motif. Such a process can be subcellular localization, secretion, post-translational modification or degradation of proteins. Conducting such studies only with experimental techniques is often associated with high costs and risks of uncertainty. Preliminary prediction of putative motifs with computational methods, them being fast and much less expensive, provides possibilities for generating hypotheses and therefore, more directed and efficient planning of experiments.

To meet this goal, I have developed HH-MOTiF – a web-based tool for *de novo* discovery of SLiMs in a set of protein sequences.

While working on the project, I have also detected patterns in sequence properties of certain SLiMs that make their *de novo* prediction easier. As some of these patterns are not yet described in the literature, I am sharing them in this thesis.

While evaluating and comparing motif prediction results, I have identified conceptual gaps in theoretical studies, as well as existing practical solutions for comparing two sets of positional data annotating the same set of biological sequences. To close this gap and to be able to carry out in-depth performance analyses of HH-MOTiF in comparison to other predictors, I have developed a corresponding statistical method, SLALOM (for StatisticaL Analysis of Locus Overlap Method). It is currently available as a standalone command line tool.

The results of the work on HH-MOTiF have been published as (Prytuliak et al., 2017). The study on the statistical method has been submitted as publication to *BMC Bioinformatics* and is currently under review.

1.2. Short linear motif definition

According to the consensus definition, a short linear motif (further - SLiM) is a stretch in the protein sequence that is required to maintain a certain function of that protein. 'Linear' in this context means that the functionality does not directly depend on the protein fold or specific tertiary structure. Many sources (e.g., (Edwards and Palopoli, 2015), (Ren et al., 2008)) add more specifically that such a function consists in interaction with a globular domain of another protein.

In literature, one can also meet other, synonymous terms for a SLiM: a motif, a pattern (Rigoutsos and Floratos, 1998) or a MoRF (Fang et al., 2013).

The provided definition implies several aspects worth discussing here, as they define the scope of this work. It also raises the question, whether one cannot formulate a more useful definition for bioinformatics applications.

First, structural motifs (in context of antibodies are also known as epitopes) are not considered as SLiMs. Structural motifs are protein surface patches consisting of structurally close residues that do not necessarily form a linear motif and which mediate binding (Kinjo and Nakamura, 2013). Strictly speaking, SLiMs are a subcase of structural motifs.

Second, SLiMs should be distinguished from conserved domains – larger units with distinct tertiary structure required for the functionality. Although there is no strictly defined border between SLiMs and conserved domains, the maximal length of SLiMs listed in ELM (Dinkel et al., 2016) – one of the most comprehensive SLiM databases – is 23 residues. Consequently, I consider longer functional units as conserved domains. At this point, I would like to point at the terminological confusion in names of some conserved domains. For example, the HTH (helix-turn-helix) domain is often referred to as HTH *motif* (Brennan and Matthews, 1989), sometimes even interchanging within a single publication (Suvorova et al., 2015). Nevertheless, its five helices span more than 70 residues (Frandsen et al., 2013) and therefore the HTH motif/domain should not be considered a SLiM.

Third, the definition of a SLiM as given by Edwards and Ren does in principle not include protein sequence stretches that have a distinct function from binding to a globular domain of another protein. If such a restriction is justified, remains an open question. The cases not included are non-exhaustively exemplified by the following:

- DNA/RNA-binding sequence elements. These technically could also count as SLiMs and therefore it seems counter-intuitive that the definition by Edwards and Ren excludes them. Although I did not find any justification behind this, I also could not find any counter examples (i.e., *short* motifs that bind to DNA or RNA). Instead, there are larger domains (with median length more than 100 residues) that are responsible for DNA/RNA binding (Malhotra and Sowdhamini, 2015). Nevertheless, short enough protein sequence units that binding to DNA/RNA may be still identified in the future. In this case, there will be a need for broadening the definition of a SLiM.
- Sequence elements with primarily non-binding function. These might be necessary for structural stability (e.g., beta roll motifs (Blenner et al., 2010)), pH optimization (e.g., C-terminal tail of tubulin (Sheldon et al., 2015)), etc.
- Sequence elements that bind to disordered (non-domain) regions of other proteins. These may be responsible for protein aggregation (Defenouillère et al., 2016) or, in theory, silencing/inhibiting a true SLiM, though I did not find any evidence in literature that such an interaction type has been described.

Fourth, the definition does not require the SLiM to be sufficient for the function (see for instance (O'Neal et al., 1995), (Milewski et al., 2001)). This aspect has the consequence that a single linear part of a structural motif or epitope can be viewed as a SLiM., If a SLiM is defined as required but not sufficient, it usually means that other parts of the protein probably facilitate the interaction but through a different mechanism. For example, the correct membrane localization of CFTR (cystic fibrosis transmembrane conductance regulator) requires the presence of more general membrane-affine regions in addition to a specific PDZ-motif (Milewski et al., 2001). Nevertheless, motifs can also be both, required and sufficient for the function (e.g., (Gasser et al., 2012)). In this case, the binding is also observed with isolated peptides (Gorelik and Davidson, 2012).

Finally, from the bioinformatics perspective, a SLiM is the aggregate of all possible sequence variants of protein sequence stretches, including those not yet detected, that still fulfill the required function: this means that the mechanism of the SLiM's ability to interact with the binding partner must be retained. In this manner, one can speak about a SLiM as a sequence space. This definition does not contradict the initial one but rather includes it. In addition, it is more practical for building SLiM predictors, as their task is to find not yet described SLiMs.

1.3. Databases of SLiMs

1.3.1. ELM

The ELM (eukaryotic linear motifs) (Dinkel et al., 2016) is currently (June 2017) the most comprehensive database of experimentally verified SLiMs. It contains 262 classes with a total of 3030 instances that correspond to distinct SLiMs in individual proteins. The ELM classes are furthermore grouped into 6 categories: CLV (cleavage), DEG (degradation), DOC (docking), LIG (ligand binding), MOD (post-translational modification), TRG (targeting).

Each ELM instance consists of a protein identifier (Uniprot-KB ID) with the start and end positions of the SLiM described. If a SLiM forms repeats in the same protein, the corresponding number of instances appear in the database. SLiMs confirmed in orthologs of the same protein in different organisms are also included. Sometimes, the instances of the same class are overlapping (e.g., the class LIG_SH3_3 in the protein Q9N2H0).

In addition, the ELM database also contains detailed descriptions of the described SLiMs, as well as primary sources of evidence for a large part of the instances. If available, PDB structures (the PDB database is available under www.rcsb.org , (Berman et al., 2000)) are also linked.

It should be noted that some ELM instances correspond to sites that were suggested *not* to be a functional SLiM. These instances are marked as 'false positive' (a total of 46 instances out of 3030), in contrast to 'true positive' instances that mark confirmed SLiMs. However, after following the associated references, I found that these instances

do not actually represent completely non-functional sites, but rather non-typical binding sites. For example, the LxCxE motif in the transcription factor Elf-1 binds the pocket domain of Rb (retinoblastoma gene product); however, the main effect of this binding results not in inhibiting Rb, as it is common for other instances in the class LIG_Rb_LxCxE_1, but in inhibiting Elf-1 itself . Furthermore, I could not follow the reasoning for marking some other SLiM instances as false positives (e.g., ELMI001864), as the corresponding sources clearly proved their functionality in the given context (e.g., CaM-binding motif in Sec61α (Erdmann et al., 2011)). In one case, namely the instance ELMI003135, the associated source (Xu et al., 2006) does not mention the SLiM in question at all.

The Table 1 summarizes the statistics on ELM composition as of March 2016. I include only classes that have instances in at least 3 proteins (the dataset that was used for testing performance of different SLiM predictors – see Section 2.4.1. Datasets used for optimization and performance evaluation).

Type	Proteins		SLiMs			Most abundant class		
	N	av. len.	N	classes	len. (av.)	name	SLiMs	proteins
CLV	64	962.1	79	6	3-7 (5.3)	CLV_C14_Caspase3-7	39	30
DEG	164	538.3	177	17	3-18 (8.1)	DEG_SCF_TIR1_1	24	24
DOC	209	708.2	285	19	2-17 (6.4)	DOC_WW_Pin1_4	96	57
LIG	918	748.8	1300	96	3-20 (7.0)	LIG_WRPW_1	95	95
MOD	301	699.4	543	21	3-12 (6.6)	MOD_N-GLC_1	156	33
TRG	187	628.5	225	17	3-23 (7.6)	TRG_ER_diArg_1	27	27

Table 1. ELM (as of 26.03.2016) composition statistics excluding classes with instances in less than 3 proteins

As one can see, the size of the types and classes are highly unequal, while the length distribution remains consistent. The CLV type is an exception, as it contains shorter SLiMs in longer proteins.

1.3.2. Other, non-specialized SLiM databases

1.3.2.1. MiniMotif Miner

Minimotif Miner (MnM) version 3.0 contains information on around 300,000 SLiM instances (Mi et al., 2012). Moreover, the MnM records contain not only the SLiM position, function, protein, and reference, but also broader information on both, the SLiM containing protein and its binding partner, as well as the interaction itself. There are in total 28 attributes to an annotated SLiM. To collect this tremendous amount of

information, the authors of MnM used the annotation helper MimoSa (Vyas et al., 2010), as well as collected information from a number of other protein databases.

MnM 3.0 contains not only the database but also a motif search engine, which scans query proteins for SLiMs available in the database. Elaborated statistical models and contextual filters remove the vast majority of false positive hits, so that the accuracy of the predictions reaches 90%.

Despite many advantages, the underlying MnM database is not available for download, which reduces its usability. Moreover, the reachability of the MnM web-server is not always given.

MnM restricts the length of motifs to less than 15 residues. From the publication, it is not clear, if longer SLiMs are discarded or trimmed (for comparison: ELM contains SLiMs up to 23 residues long – see Table 1).

1.3.2.2. PROSITE

PROSITE is a database of patterns (regular expressions) that correspond to experimentally proven domains, SLiMs, and other functional sites. PROSITE does not provide distinctions between SLiMs and non-SLiMs; it annotates, among others, also motifs that maintain the 3-dimensional protein structure. If a given motif has several known functions, they are all described by indicating the role of individual conserved residues. The release of 10.05.2017 contains 1787 entries with 1309 associated patterns. PROSITE also implements a web application for scanning query proteins for PROSITE patterns in database.

The PROSITE entries are available for download. However, they contain only general information on the patterns, without details on individual instances.

1.3.3. Specialized SLiM databases

1.3.3.1. iLIR database

The iLIR database (Jacomin et al., 2016) contains the positions of all known instances of the LIR (LC3-interacting region) motif. This motif is responsible for the interaction with

Atg8-family proteins, which are required for autophagy-dependent degradation of their substrates. The database as of 02.06.2017 contains 6086 records in 3495 different human proteins. In addition, it contains data for 7 widely used model organisms.

All the data from the iLIR database, including the motif positions, are available for download as a CSV file.

1.3.3.2. Scansite

Scansite (Obenauer et al., 2003) is the database for phosphorylation motifs. Version 4.0 consists of 70 mammalian and 54 yeast motifs, which are further grouped depending on the kinase type associated with a phospho-site.

Like other databases, Scansite comes with a web application to scan user-provided query sequences for available phosphorylation motifs. The service provides also additional types of calculations, which may supplement the SLiM search, such as surface accessibility, amino acid composition, etc.

Scansite 4.0 allows viewing the sequence logos of the individual phospho-motifs through the web interface, but unfortunately it does not provide the possibility to download full records as a file.

1.3.3.3. PhosphoSitePlus

PhoshoSitePlus, or PSP (PhosphoSitePlus(R), www.phosphosite.org) (Hornbeck et al., 2015), is a protein modification resource, which contains information on post-translational modifications including phosphorylation, methylation, ubiquitination, succinylation, etc. As of 17.05.2017, PSP contains information on 516,641 records in 53,577 proteins (of these, 20,264 are non-redundant). In addition, this resource provides linkage information of specific PTMs to genetic mutations and diseases. PSP also has a dedicated Cytoscape (Su et al., 2014) plugin.

All the sites and datasets from PSP, including positions of modified residues are freely available as TSV files; however, the downloading requires user registration and login.

1.4. Representation of SLiMs

1.4.1. Regular expressions

Regular expressions (shortened singular form: regex) as concept were developed by the mathematician Stephen Cole Kleene in the 1950s (Kleene et al., 1956). They represent a way to describe a search pattern in a text string or document, if a search result must not necessarily be a fixed substring but rather a range of possible substrings satisfying certain criteria. In general, a regex is the shortest way to write such a pattern.

Regexes always make use of some set of special characters (thus being a kind of metalanguage). Occurrence of these characters in a pattern may have two meanings: (1) literal occurrence of this character in the searched text, and (2) its special meaning. For example, a dot may mean a literal dot as well as any character. Depending on the exact application, the special character set may vary. The most common standards are the POSIX and the Perl syntaxes. A much narrower standard, but following the same principle is the IUPAC nucleotide degenerate code, in which different letters specify all possible combinations of nucleotides allowed at a given position. Similar limited code exists for proteins (in this code, Z, for example, means either Glu or Gln). PROSITE patterns are a standard developed to describe possible amino acid combinations in a motif. It specifies a syntax for lists of possible and prohibited amino acids at a given position as well as gaps of variable length. The reference is available under http://www.hpa-bioinfotools.org.uk/ps_scan/PS_SCAN_PATTERN_SYNTAX.html.
Nevertheless, many biologists and programmers tend to use Perl regexes to describe SLiMs instead. One of the prominent examples is the ELM database. I will also be using the Perl syntax throughout this work. The important syntax elements are exemplified in the Table 2. For the full reference, visit http://perldoc.perl.org/perlreref.html.

Syntax	Meaning
`.`	any single symbol
`*`	the preceding symbol repeated any (perhaps, zero) number of times
`.*`	arbitrary string (incl. empty string)
`?`	the preceding symbol repeated exactly either zero or one times
`A?`	either single letter A or empty string
`{N}`	the preceding symbol repeated exactly N times
`.{2}`	exactly two arbitrary symbols
`{K,M}`	the preceding symbol repeated at least K but no more than M times
`A{2,4}`	one of the following strings: AA, AAA, AAAA
`[]`	exactly one symbol of the listed inside the square brackets

Table 2. Selected Perl regular expression syntax examples

The choice of syntax may not only be a matter of convenience – Perl regexes are generally shorter than PROSITE patterns – but also of functionality: Perl allows for patterns that cannot be described with a PROSITE pattern. In particular, this is based on usage of pipes (`|`), which allow listing several exact combinations for stretches longer than 1 residue. For example, let us consider the two following hypothetical SLiMs:

SLiM 1: `AA`
SLiM 2: `CC`

One potentially can use a Perl regex `(AA|CC)` to match exactly these two. However, excessive usage of pipes will lead to very long and incomprehensive regexes. It will also undermine the very purpose of using regexes instead of plain lists of occurrences. Therefore, in order not to break the PROSITE convention, one can use a more limited `[AC][AC]` here, although it additionally matches `AC` and `CA`. This constraint is desirable, if one wants to enumerate regexes combinatorially with relatively straightforward algorithms (see examples in Section 1.5.3.2. *De novo* predictors). It also keeps regexes easily readable and reasonably short. However, it is not always used. For example, the ELM class TRG_ER_diArg_1 is described by the following regex:

```
([LIVMFYWPR]R[^YFWDE]{0,1}R)|(R[^YFWDE]{0,1}R[LIVMFYWPR])
```

The important property of regexes is their ability to nest: this means that one regex can include another one. Let us as an example consider the following three strings:

String 1: `ACD`
String 2: `AACD`
String 3: `ACE`

There are a lot of possible regexes that can describe these strings. A few examples are listed below:

```
A.*
A.*C.*
A.*C.
A.?C.
```

```
A.?C[DEF]
A.?C[DE]
A{1,2}C[DE]
```

Each of this regexes matches all of the three strings above. However, the latter three are more precise, less ambiguous representations. The last variant (or its equivalent `AA?C[DE]`) will match only four strings in total (it matches `AACE` in addition), while the three on top will match infinite numbers of potential strings. It is also true that a lower regex in this list is nested in an upper (i.e., an upper will match all the strings the lower does). When describing a SLiM, it is important to choose the least ambiguous regex possible.

1.4.2. Profiles

A motif profile is a way to store information about a motif that includes frequencies of observed amino acids and in some cases gaps for each relevant position. One can see them as quantitative multiple sequence alignments (MSAs).

A common way to represent a profile is a position-specific scoring matrix (PSSM), which is sometimes also called position weight matrix (PWM). The concept of PSSMs for sequences was introduced in 1982 in the course of a study on bacterial proteomes (Stormo et al., 1982). Columns of a PSSM correspond to sequence positions and rows represent all possible amino acids.

Unlike regexes, which solely list possible amino acids for each position, profiles also specify how often each of them might be observed. Nevertheless, profiles do not necessary provide information on individual observed instances. For example, let us consider the following PWM:

Residue	Column number		
type	1	2	3
A	1	2	1
C	2	1	2
…	0	0	0

It could have been formed by the following three sequences:

```
AAA
```

```
CAC
CCC
```

However, the following three also do match:

```
ACC
CAA
CAC
```

Thus, conversion of an MSA to a profile is lossy by nature. Because of this, the profiles may not capture sequence diversity on a deep enough level and still be misleading if, for example, motif instances are divided into subtypes.

Another form of a profile is the sequence logo (Schneider and Stephens, 1990) – a graphical representation, in which each position corresponds to a letter or a set of letters. The height of the letters usually corresponds to the enrichment of the corresponding amino acid relative to its background frequency at a given position. The enrichment is usually represented as information content (Schneider et al., 1986) measured in bits, although other variants have been proposed (e.g., iceLogos (Colaert et al., 2009) plot fold changes by default).

Taken together, matrices are a convenient way to represent sequence profiles for further computer processing, while logos are good for their visualization.

1.4.3. Hidden Markov models

Hidden Markov models (HMM) resemble classical PSSMs. However, in addition to amino acid frequencies, they also comprehensively describe the gap patterns in the underlying MSA.

The basic idea behind hidden Markov models is that a sequence of unobservable (hidden) events causes another, observable, sequence of events. HMMs were first described in 1966 (Baum and Petrie, 1966). Since then, it found multiple applications. HMMs were used for sequence alignments since the 1990s (Eddy, 1996). In this context, the implicit assumption is made that the sequences aligned have a common ancestor, while the HMM itself encodes the differences throughout the phylogenetic tree. These

differences form the mentioned hidden states, while the sequences themselves are observed.

HMM profiles are at the core of the novel alignment method `hhsearch` (Söding, 2005) that together with other HMM-based applications is part of the HH-suite (Remmert et al., 2011) (see more details in Section 1.7.3. HH-suite).

The key practical difference from the PSSMs is that HMM profiles provide so-called transition probabilities for each position in the sequence. These probabilities are, in point of fact, frequencies of gap openings and gap closings at the given sequence position. There are two types of gaps in a sequence profile HMM: insertions and deletions. Insertions denote the positions that are absent in the ancestor sequence but appear in the branches; deletions are the opposite. Such comprehensive gap handling allows HMM-based methods to consistently outperform other methods for motif search and homology deduction in case of low sequence conservation (Meier and Söding, 2015) (Siebert and Söding, 2016).

1.5. SLiM discovery techniques

1.5.1. Direct experimental methods

The binding capacity of a SLiM to a target protein can be measured experimentally. Among the most popular techniques of experimentally verifying a binding event is to use immunoprecipitation – western (IP-Western) techniques (Renart et al., 1979)(Taylor and Posch, 2014). Other techniques include isothermal titration calorimetry (Pierce et al., 1999) and solution NMR (Liu et al., 2016).

Deletion studies or the engineering of chimeric proteins are methods to identify putative interacting regions for a functional SLiM without prior knowledge. Whenever a SLiM is tested for being required and sufficient, researchers have to use mutants of the putative important, interacting residues. The examples of SLiM identification studies applying deletions are (Gan et al., 2000), (Deretic et al., 1998). The examples of studies constructing chimeric proteins for these purposes are (Oeste et al., 2014), (Foss et al., 2013). Intriguingly, the naturally synthesized chimeric proteins (those containing parts

encoded by different genes) also tend to swap the motifs and domains between the original proteins (Frenkel-Morgenstern and Valencia, 2012).

X-ray crystallography is the most direct but also the most laborious method to study protein interactions (Turnbull and Emsley, 2013). The main advantage of obtaining a complex structure is that it gives direct information about the specific residues, which are involved in the interaction; moreover, it also informs us about the mechanism, how the interaction is formed (e.g., through hydrogen bond formation or through residue stacking). This practically eliminates the need of generating multiple mutants or chimeras to guess the key interacting residues. Calculation of the resulting interaction strength (i.e., the absolute free energy) from the structure is not straightforward (Aldeghi et al., 2016). Nevertheless, the main hindrances for the widespread application of X-ray crystallography are its low success rate, laboriousness, and high cost. As of 12.06.2017, there are only 15,781 protein complexes deposited in PDB (Rose et al., 2015). At least 9,801 of these represent binary hetero-dimers (AB, A2B2, A3B3, and A2B types). However, according to the STRING database (Szklarczyk et al., 2015), there are more than one billion protein-protein interactions. Thus, it is not feasible to crystallize all possible interaction pairs to establish the nature of all possibly existing interactions.

Specific pipelines for SLiM discovery, which include bioinformatics as well as experimental methods, were described in (Gibson et al., 2015). From this study, it is clear that it is important to carefully consider the motif identification strategy to get the unambiguous answer on motif presence and the key residues of a functional SLiM.

1.5.2. Main concepts of computational SLiM predictors

A computational SLiM search can be broken into three distinct logical steps: pre-filtering, matching (the search itself), and post-filtering, the first and the third being optional.

Pre-filtering consists of excluding regions in the protein sequence that that are very unlikely to harbor the SLiM of interest. This process is performed on the basis of characteristic features of the protein sequence itself, irrespective of sequence similarity to the given input motif. These features can mostly be classified into one of the two

categories: protein properties and sequence properties. Many tools also include a homology filter that removes or clusters too similar sequences from the input set.

The most relevant properties for motif discovery are molecular function, cellular function, and subcellular localization. The importance of these properties is emphasized in the description of the MiniMotif Miner motif searching engine (Mi et al., 2012), where they are implemented as contextual pre-filters. This type of pre-filtering always excludes whole protein sequences.

Pre-filtering on the basis of sequence properties, on the other hand, excludes only specific sequence stretches, although these can in rare cases span over an entire protein. Therefore, this type of pre-filtering is also known as masking. Features like surface exposure, disorder, low complexity or evolutionary conservation are frequently used properties for masking. In addition, it is common to check for the presence of known conserved domains, as it is unlikely for a SLiM to be situated in such a domain (Neduva et al., 2005).

In addition to the pre-filtering principles described above, the user can also do pre-filtering manually. By providing a specific protein set as input, the user in fact already performs pre-filtering by protein properties, though not including undesired proteins in the set. Furthermore, the majority of SLiM predictors allow the user to provide a custom mask defining selected stretches in the input sequences. The manual and automatic pre-filtering can be combined when searching for a motif.

The matching is essentially the 'core' of any motif searching method. At this step, the matching SLiM candidates are found in the unmasked regions of the retained proteins. The output of the search algorithm is a list of SLiM candidates (hits) with associated numerical scores. Algorithm may be stopped after finding a specified amount of hits, or limited to return only hits above or below a certain given score threshold. Such filtering by the alignment score should not be confused with the post-filtering described later in this section.

The word ‘matching’ in this context means establishing a set of unequivocal matches between some residues in the pair of sequences without violating the original sequential order. In other words, a set of sequence alignments is produced at this step, although the algorithm may be quite different from classical sequence alignment techniques (see below). These alignments are local and can contain gaps in the aligned sequences. Following the concept of SLiM degeneracy, several different instances of the same SLiM candidate generally must not necessary match in every position, as only few columns in the alignment may be conserved. SLiM alignments produced at the matching step are typically pairwise and not multiple in nature. This holds true even if the input motif and/or the query are actually profiles based on MSAs of many sequences. In this case, the participating profiles are aligned to each other ‘as is’, without being reassembled or corrected. This can also be approximated as a pairwise alignment of two consensus sequences.

Matching methods in SLiM predictors do not incorporate traditional sequence search techniques like BLAST (Altschul et al., 1990) or alignment engines like Clustal (Sievers et al., 2011) or MAFFT (Katoh et al., 2002). In fact, such methods are not suitable for searching or aligning SLiMs due to their shortness and their poor conservation. Motif matching can in in this context also be one of the techniques that are termed ‘alignment-free sequence comparisons’ (Bonham-Carter et al., 2014).

Obtained alignments of SLiM candidates typically have a numerical value associated with them, such as a significance value, an e-value, etc.. This value is hereafter referred to as the score. This allows to: 1) rank the candidates. Only the very best candidate or certain number of the top ones is then selected for further consideration; 2) threshold the candidates. Only the ones that exceed some predefined score threshold are selected; as a result, the selection may also be empty.

Post-filtering consists in additional checks imposed on the identified alignments not directly connected to the alignment quality itself as described by the score. This may compensate for shortcomings of the matching algorithm, as well as incorporate additional information about the sought-for SLiM(s). The shortcomings in this context mean associating better scores with seemingly worse candidates: as an example, if raw

BLAST scores were used, conserved domain and homology matches would overshadow the putative SLiMs. In such a case, good results can be obtained neither by ranking nor by thresholding.

When certain shortcomings of a reasonably good matching algorithm are identified, it is sometimes easier to introduce an additional step – a post-filter – to modify the scores of the proposed SLiM candidates or ultimately to remove some candidates completely than to re-write the whole algorithm from scratch in order to achieve better scoring. Consequently, some novel predictors are built on well-established methods that use sub-optimal matching algorithms. In this case, some improvements to the scoring approach via implementing appropriate post-filters were implemented. An example of such a tool is SLiMDisc (Davey et al., 2006), which is built upon TEIRESIAS (Rigoutsos and Floratos, 1998).

Possible shortcomings include for instance insufficient treatment of gap penalties for aligning degenerate SLiMs or inadequate consideration of the contextual background amino acid frequency. Post-filtering consists in this case in introduction some kind of a score correction term. This term is generic in nature, i.e., applicable to the vast majority of SLiMs. For example, one can discard too short hits or matches in only abundant amino acid types, even if the amino acid match is perfect and has passed the optimal score threshold.

Some additional information may be given about a SLiM. This usually includes information on key residues, perhaps in form of a table that lists integral residues and their possible variants. This acknowledges the fact that although a SLiM may be degenerate, some residues must remain invariant to preserve the motif function. A typical example is the SUMOylation site. It has many variants (Endter et al., 2001) (Diella et al., 2009) (Hietakangas et al., 2006), which must be considered separately while developing a predictor for this motif (Beauclair et al., 2015). Nevertheless, a SUMOylation site always has to contain a Lysine to be SUMOylated. Thus, a high-scored alignment hit with well matching flanking amino acids and an Arginine instead of the Lysine must be discarded at the post-filtering stage. Another good example is the ELM motif LIG_AP_GAE_1 (Duncan et al., 2003) (Mattera et al., 2004) (Lui et al., 2003)

containing a number of acidic residues. It can be aligned with a high score to a D/E low complexity region by most algorithms. However, the motif is not functional without an aromatic residue in the middle. Thus, introducing a simple check for an aromatic residue may significantly reduce the false positive count, while searching for this motif.

1.5.3. Classification of computational SLiM predictors

1.5.3.1. Template-based SLiM predictiors

In some cases, the motif is already known and the task is to find the same motif in previously unconsidered proteins. Such a task is a conceptually straightforward problem. It requires some sort of scanning of query protein sequences for stretches that are similar to the input motif.

There are two basic scenarios, when the template-based prediction is suitable:

1. Search for a specific SLiM to identify novel proteins with the same function.
2. Search for all possible SLiMs in a given protein.

Scenario 1 requires a SLiM as an input. A search for SLiMs not present in databases is also possible. The target proteins can be predefined or also provided by the user. A typical example of a template-based predictor is SLiMSearch (Davey et al., 2011).

Scenario 2 can be satisfied through built-in database predictors. A template-based SLiM predictor is a natural extension of a database of SLiMs. Almost all databases discussed in this work have built-in predictors of the motifs present in the database in user-provided query sequences. Some of them, for example, MiniMotif Miner, employ elaborated algorithms to ensure the prediction accuracy. Nevertheless, as they are only an addition to a database, they often provide solely basic functionality, which is not sufficient for all possible needs of the user. For example, the ELM motif predictor accepts only single protein sequence and does not provide a convenient API. Therefore, specialized services for searching for motifs from the popular databases were developed. A prominent example is the iELM service (Weatheritt et al., 2012), which contains pre-compiled hidden Markov model profiles of ELM motifs for searching in the user queries, both, in single sequences, as well as in interacting networks.

1.5.3.2. De novo SLiM predictors

In contrast to the template-based prediction, a *de novo* motif search does not require any a priori knowledge about the SLiM(s) to be detected. Instead, it looks for overrepresented sequence stretches that have a low enough probability to appear purely by chance. This probability is typically expressed in form of a p-value. Necessary pre- and post-filtering steps are performed to selectively remove false positive SLiM candidates. The core of a *de novo* predictor, its search algorithm, consists of two parts: a SLiM candidate generator and a statistical model that calculates the corresponding p-values, which may be used as scores for ranking of the candidates.

Here I illustrate the workflows of a SLiM candidate generator with two examples: SLiMFinder (Edwards et al., 2007) and GLAM2 (Frith et al., 2008).

SLiMFinder employs a dedicated algorithm for this purpose - SLiMBuild. SLiMBuild enumerates all possible regular expressions present in the input dataset within defined constraints: minimal and maximal number of non-wildcard positions, maximal length of wildcard stretches, etc. The sequence start and the sequence end are treated as separate amino acids. If there are nested regexes that match stretches in the same number of sequences, only the least ambiguous ones are retained. If a pair of regexes can be merged into a more ambiguous regex to increase the number of occurrences in non-homologous sequences, such a merge is performed to produce a new SLiM candidate. Simply put, the SLiMBuild algorithm generates all possible amino acid regular expressions and subsequently discards those that are not present in a sufficient number of the input sequences. The actual implementation is of course more efficient than this.

GLAM2 employs a simulated annealing algorithm (Kim et al., 1994) to optimize alignments between possible instances of the same SLiM. This algorithm is a variant of the Gibbs sampling algorithm (Neuwald et al., 1995). It starts with an initial, possibly non-optimal local alignment (i.e., SLiM candidate) and tries to iteratively improve it by introducing random changes. These include adding and removing of putative key columns (‘column sampling’) as well as adjusting individual sequences (‘site sampling’). The iterations are carried out until convergence. Repeating the procedure several times ensures that the algorithm does not get trapped in a local optimum and that different

SLiM candidates are detected in the same set of sequences. Although in theory the output of GLAM2 is random to a certain degree, I did not observe discrepancies between several runs on the same data with the same parameters. This indicates the robustness of the GLAM2 simulated annealing algorithm.

All generated SLiM candidates are then subjected to the corresponding statistical model implementation that assigns p-values on the basis of the regex/alignment properties. These include the number of sequences with occurrences, the length of the SLiM, its ambiguity, as well as background amino acid frequencies. The statistical model of SLiMFinder is discussed in more detail later in this thesis.

1.5.3.3. Discriminative de novo SLiM predictors

A discriminative motif predictor finds SLiM candidates that are overrepresented in a positive protein set in comparison to a negative, set. The negative set is usually chosen in a way that it theoretically does not contain the SLiM(s) of interest. For example, for a task of predicting mitochondrial targeting signals (Neupert and Herrmann, 2007), a set of nuclear proteins can serve as a negative set. A classical example of a SLiM predictor that employs negative sets on the basis of subcellular localization information is presented in (Lin et al., 2011). Alternatively, the negative set can be replaced with a background set. The background set theoretically contains the sought-for SLiM(s) but with significantly lower frequency in comparison to the positive set. The background set is normally a superset of the positive set. A typical choice for the background set is the whole proteome of the species being studied.

The pipeline of a discriminative SLiM predictor adds one additional step to what would otherwise be a classical *de novo* predictor. This step consists in searching for the SLiM candidate identified in the positive set in the negative one. A statistical score, calculated on the basis of the identified numbers of occurrences, is assigned upon here. Therefore, a discriminative SLiM predictor can also be viewed as a *de novo* predictor with a specific statistical model or an extra post-filtering step. Nevertheless, I define it as a separate type of predictors, because it requires different input data in comparison to a classical *de novo* predcitor. It was shown in (Song et al., 2015) that adding a discrimination

capability to a SLiM predictor can in general only marginally improve the resulting performance.

The obvious weakness of discriminative algorithms is the necessity to compile a suitable negative set. While it may be obvious in some cases, there is usually not enough information in a typical SLiM search to reliably mark a large group of proteins as being devoid of a particular SLiM. For example, while studying Golgi sorting and trafficking (Crevenna et al., 2016), one cannot easily exclude proteins with certain functions or localizations, because various proteins are modified in the Golgi apparatus before being transported elsewhere. Even finding a reasonable background set can be problematic, if the input dataset contains proteins from different species, which is often reasonable, as the majority of ELM classes do contain instances in different species.

A general example of a discriminative SLiM predictor is MotifHound (Kelil et al., 2014). MotifHound is not restricted to subcellular localization datasets. As input, it accepts the background set, which is mandatorily a superset of the positive set. The logic of SLiM candidate generation in MotifHound is similar to that of SLiMBuild. After generating the SLiM candidates, it performs the hypergeometric test to assign the corresponding p-value $\boldsymbol{p}$ as the probability of a random occurrence in at least $\boldsymbol{k}$ sequences in the positive set of size $\boldsymbol{s}$, while there are occurrences in $\boldsymbol{n}$ sequences in the background set of size $\boldsymbol{b}$:

$$p = \sum_{i=k}^{\min(n,s)} \frac{\binom{n}{i}\binom{b-n}{s-i}}{\binom{b}{s}}$$

Thus, the statistical score of a MotifHound SLiM candidate, unlike the one from SLiMFinder, is not dependent on the properties of the SLiM itself.

The statistical model of MotifHound demonstrates another integral weakness of discriminative SLiM predictors: the score depends on the relative set sizes. Moreover, it can change dramatically upon small positive set contaminations. Let us consider a SLiM with the following properties: it occurs in 50% of proteins with the function X, while it occurs only in 20% of proteins without the function X. Note: the reasons, why there may be so many false positive occurrences of a real SLiM are discussed in (Gibson et al.,

2015). Let us further assume that MotifHound detects this SLiM with perfect accuracy. Table 3 shows the resulting p-values depending on different positive and background set sizes. As one can see, the p-value is heavily dependent on the set sizes. For example, in an organism with 10,000 proteins, a SLiM predicted in a subset of 100 proteins will have a p-value that is 5 orders of magnitude lower than other SLiMs with the same underlying statistical properties predicted in a subset of only 50 proteins. Moreover, such discrepancies can be caused not only by changes in the real size of the positive set but also by failure to identify the set precisely. For example, let us assume that the positive set identification was not perfect and as a result, it included 10 additional non-related proteins (see row 8 of Table 3). This relatively small contamination will actually worsen the p-value approximately 7-fold in contrast to the p-value of the non-contaminated positive set of 100 proteins and approx. 60-fold in contrast to the same set containing 110 proteins. This 60-fold change is just the effect of the positive set contamination and again not the properties of the underlying SLiM. This change would be even more drastic, if the difference between background frequencies of the SLiM in positive and negative sets were bigger (e.g., 80% and 5% instead of 50% and 20% respectively). Taken together, this means that, at least without appropriate corrections, p-values based on the hypergeometric test may be practically only used for SLiM ranking within the same search, and not for comparing of SLiM strength between searches on different data.

No.	s	b	k=s/2	n=k+(b-s)/5	p
1	10	20	5	7	.029
2	10	100	5	23	.0090
3	10	1,000	5	203	.0066
4	10	10,000	5	2,003	.0064
5	50	10,000	25	2,015	5.3e-7
6	100	10,000	50	2,030	6.9e-12
7	110	10,000	55	2,033	7.6e-13
8	110*	10,000	52*	2,030*	4.5e-11

Table 3. p-values assigned by MotifHound for a hypothetical SLiM prediction.
* The values are not derived using the formulas. For explanations, see the text

The two described weaknesses of discriminative tools, namely the difficulty to determine the meaningful negative/background set and the strong dependency of the results on the set sizes and small contaminations, render them less practical for application on real data.

1.6. Machine learning in template-based SLiM prediction

1.6.1. Learning based on sequence features only

The common weakness of both, classical alignment approaches and *de novo* regex construction such as is done in SLiMBuild is their inability to capture transient relationships between different positions (also referred to as columns) of a putative motif. These methods are limited to considering amino acid frequencies within single columns, as well as determining, which columns will be retained. As an example, let us consider the hypothetical motif A in Table 4. The six sequences already form a quite good MSA. They can be summarized by the regex `[KR][ILV].W[DE]`. However, neither an MSA algorithm nor SLiMBuild grants a score improvement because of the fact that `K` in the first position is always coupled with `E` in the last position, while `R` is always coupled with `D`. Hidden Markov models are generally more flexible. The utility `hhmake`, which is part of HH-suite and generates the HMM from an input MSA, indeed produces different HMMs from the given first set and the second set, in which the occurrences of Asp and Glu in the last column are flipped (motif B in Table 4). This difference leads to a slightly higher score of HMM profile-to-profile comparison against the target motif HMM for motif A (see Table 4), which has the same `K/R-D/E` coupling pattern. Nevertheless, this difference is quite small in comparison with other factors influencing the score: the Viterbi scores are 10.60 and 10.55, respectively with the autocorrelation term turned off in `hhalign`,. Moreover, HH-suite remains a general-purpose alignment tool, albeit with broad functionality and specifically designed for weakly conserved sequences. Therefore, it does not conduct in-depth statistical analysis of amino acid composition in a pair of compared profiles.

Motif A	Motif B	Target motif
RIGWD	RIGWE	KVHWE
KLMWE	KLMWD	RLAWD
KISWE	KISWD	RVNWD
RVQWD	RVQWE	KIGWE
KVHWE	KVHWD	RIPWD
RLPWD	RLPWE	KLTWE

Table 4. Selected hypothetical motifs to illustrate inter-column residue dependencies

Machine learning (Baştanlar and Özuysal, 2014) is a much broader approach for detecting statistically significant patterns in input data. A properly configured machine learning pipeline can potentially detect all significant patterns of inter-column

composition dependencies within the considered sequence window. In comparison to alignment-based approaches, this would deliver more efficient solutions for two distinct problems: 1) the search for a given weakly conserved SLiM within a target sequence; 2) the detection of 'SLiM-like' regions within a given sequence, which defines a search for SLiMs of all types.

Here I provide two examples of a successful application of machine learning for tackling the first problem. These examples have one common feature: they both contain a contingency table of dependencies between all possible pairs of columns within the input window. Such a table is filled with values of a statistical measure that shows, how likely the observed coincidences in amino acid composition can arise by chance.

In the first example, O-GlcNAcylation sites classified into subtypes were detected in an out-of-sample test set with 84% accuracy (Kao et al., 2015). In this study, the contingency table was used to perform the classification with subsequent generation of HMMs for each determined subtype. The algorithm for classification on the basis of the contingency table is known as maximal dependence decomposition (MDD) and was first implemented in a tool called MDDLogo (Lee et al., 2011). The parameters for HMM generation were optimized so that the employed support vector machine (SVM) (Schölkopf and Smola, 2001) classifier achieves maximal performance in binary classification of positive and negative sequence windows by the associated bit score of the HMM profile comparison. A similar approach was used by the same authors to detect ubiquitination sites with 73.7% accuracy (Huang et al., 2016).

In the second example, a machine learning framework was built to distinguish DNA sequences that fluoresce upon binding to silver cations (Copp et al., 2014). The positive set consisted of 30% of the brightest sequences (determined in the experimental setup), while the negative set contained the 30% darkest sequences. The 40% sequences in the middle range were excluded form further consideration. The framework included comprehensive enumeration of all possible regexes that contained flexible numbers of wildcard positions between conserved DNA bases but considered no degeneration in the bases themselves (i.e., without Perl square bracket syntax). The number of occurrences of each regex was turned into a feature. A feature was considered informative, if its

average value – and thus the frequency of the underlying regex – was lower than the specified threshold T_N in the negative/positive set and higher than the specified threshold T_P in the positive/negative set respectively. Thus, features enriched in either the positive or negative set were considered. After pruning of non-informative features, the SVM classifier was trained to distinguish between positive and negative cases. Augmented with some additional features, the resulting classifier demonstrated 76% accuracy on the test data.

There are also successful applications of machine learning to tackle the second problem. For example, an accuracy exceeding 70% can be reached by analyzing compositional skews and evolutionary conservation patterns in collections of orthologs (Fang et al., 2013). In this study, the protein sequences were divided into SLiM, SLiM-flanking, and non-SLiM regions. It was shown that SLiM and SLiM-flanking regions show transient but still detectable differences in composition from non-SLiM regions.

1.6.2. Learning based on diverse properties

The previous section described, how machine learning could be used to detect remote similarity in sequence stretches, going beyond the possibilities of conventional alignment techniques. Nevertheless, a comprehensive sequence analysis is not the only strength of machine learning. Its other innate advantage is being a universal method to combine information of different types and from diverse sources to make better classifications or value predictions. These other types of information include, in addition to the sequence data, structural data, biophysical properties, functional annotations, etc.

The idea of predicting SLiMs on the basis of different information sources has been realized in different forms, even if it was not explicitly called machine learning. For example, MiniMotif Miner 3.0 represents a framework for a template-based SLiM search with filters including molecular function, cellular function, and subcellular localization.

Furthermore, many tools, although not being strictly machine-learning approaches, combine diverse features to deliver more accurate results. They for instance use structural features to improve the resulting multiple sequence alignment, which are so-called posterior probability-based MSA methods. Some of them, e.g., 3DCoffee (Poirot et

al., 2004), use real 3D structures, while others, e.g., MSACompro (Deng and Cheng, 2014a), predict the structure on the basis of the sequence. The latter are of course pure sequence-based tools, although they use long-range similarity features to facilitate possible local alignments. HH-suite also incorporates the secondary structure (real or predicted) in its scoring system to produce more accurate alignments of weakly conserved sequences. Nevertheless, HH-suite, as well as other alignment methods, remain general tools and are by themselves not suited for SLiM discovery.

Using biophysical properties is another way to supplement purely alignment-based SLiM predictions. Biophysical properties of individual amino acids are already included in alignment methods, as the properties of separate amino acids, such as size, charge, and hydrophobicity are the basis for the similarity matrices, which are the basis for alignments. The most popular matrix is BLOSUM62 (Eddy, 2004). However, it may happen that some other properties are more important for the functioning of a particular SLiM. To address this possibility, one could potentially adjust the BLOSUM62 matrix (Mills and Pearson, 2013) (Hess et al., 2016) for each individual SLiM. A more intuitive solution, however, is to use machine learning to analyze the impact of all listed properties at once using the standard methods to reduce dimensionality. A good example of such a study is (Kumaran Nair et al., 2012), which not only demonstrated a good accuracy but also compared different dimensionality reduction methods, as well as different classifiers. The weak points of this study, however, were the focus on a single motif type and a lack of comparison with conventional alignment methods. Another study (Karaçali, 2012) demonstrated the superiority of a similar machine learning approach in predicting N-glycosylation sites over other available predictors. Unfortunately, it also focused on a single motif type. The common feature of these two proposed methods is using only residue-level properties, which in fact makes them sequence-only approaches.

To my best knowledge, there is currently no publications describing machine learning frameworks for SLiM prediction, which explicitly use both, sequence features and a kind of truly non-sequence information to build the initial set of features.

1.7. Selected computational tools used in SLiM predictors

1.7.1. BLAST

BLAST (Altschul et al., 1990), although being 'local', is not sensitive enough to find SLiM alignments. Nevertheless, it is used in several SLiM predictors, including SLiMFinder and HH-MOTiF. In both tools, BLAST is implemented to detect homology relationships between proteins. For this purpose, the BLAST alignments themselves are discarded and only their statistics (e-value, length, identity counts) are considered. The BLAST e-value reflects the strength of the best possible local alignment between a pair of given proteins. If certain conditions are satisfied, the proteins are considered homologous, otherwise unrelated.

Homologous relationships detected by BLAST can be used in two distinct ways in a SLiM predictor: they can be used to 1) find putative orthologs of a query protein in order to estimate residue-wise evolutionary conservation, as it was done in HH-MOTiF, which was already proven to be helpful in SLiM searches (Davey et al., 2009); and 2) find related proteins within the input set to obtain a set of unrelated proteins for further SLiM search; it is applied as such in SLiMFinder.

The BLAST algorithm was realized in many software tools and web-servers. The most popular implementation is the one from the National Center for Biotechnology Information (NCBI). The pre-installed standalone NCBI BLAST is a prerequisite for running both, HH-MOTiF and SLiMFinder.

1.7.2. MAFFT

MAFFT (Katoh et al., 2002) is an algorithm for global multiple sequence alignments. As encoded in the name, MAFFT uses fast Fourier transforms (FFT) in its algorithm with the purpose to calculate frequencies of residues of different polarity and size. The key advantage of MAFFT is its two-order of magnitude speed gain while retaining comparable accuracy in comparison to previously existing methods, such as CLUSTALW (Chenna et al., 2003) and T-COFFEE (Notredame et al., 2000). This advantage was achieved by implementing a two-step procedure: pre-locating potential homology regions with subsequent alignment of regions in between. In the following, multiple improvements were introduced to MAFFT by the developers to increase its accuracy, as well as to add new features. The current version is 7 (as of June 2017) (Katoh and

Standley, 2013). Recently added useful features include the possibility to combine existing alignments in different ways, depending on the underlying phylogenetic tree.

As a generally global alignment tool, MAFFT is not directly suited for motif searches. However, it can be used for auxiliary purposes, for example to construct an MSA from the orthologs that were identified using BLAST, as it is applied in HH-MOTiF.

1.7.3. HH-suite

HH-suite is the package for generation and alignment of hidden Markov model profiles to detect remote sequence similarity. HH-suite consists of several utilities. Among others, it contains `hhblits` (Remmert et al., 2011), `hhsearch` (Söding, 2005), and `hhalign` to perform alignments, as well as `hhmake` to generate HMM profiles from MSAs. By the scope of functions, HH-suite can be regarded as a substitution for BLAST.

The utilities used in HH-MOTiF are `hhmake`, `hhalign`, and `hhsearch`.

`hhmake` takes as input an MSA (for example, a FASTA file) and constructs a HMM out of it (as an *.hhm file). It also can do filtering of the input alignment by removing too close or too distant sequences. As the MSA normally contains gaps, it is important to decide, how many gaps in a given column are tolerable to still count the column as being in a match state. Alternatively, the match states can be assigned according to the first sequence in the MSA. All other columns will be assigned to either a deletion or insertion state.

`hhalign` takes as input two HMMs and outputs a list of hits, where each hit represents a local profile-to-profile alignment of two segments of the input HMMs. The number of hits is limited by a score threshold, as well as by the maximal number of hits (only 1 hit by default). The two HMMs are treated equally, despite the formal distinction into query and template. A typical `hhalign` hit is shown below.

```
Probab=0.21     E-value=0.29     Score=15.47     Aligned_cols=11
Identities=36%  Similarity=0.987  Sum_probs=5.5

Q first_HHM       175 KQVCTDINECE  185 (757)
Q Consensus       175 kq~C~d~neC~  185 (757)
```

```
                     ++.|.|..-||
T Consensus         70 RE~CeDy~pCe   80 (103)
T second_HHM        70 REACDDYRLCE   80 (103)
Confidence             33555555554
```

`hhsearch` is similar to `hhalign`; however, it can compare one query with a library of templates. The advantages over `hhalign` are only semantics and speed.

1.8. Challenges of computational SLiM discovery

1.8.1. Low conservation of SLiMs and their statistical (in-)significance

The main problem with searching SLiMs *de novo* is their low conservation. For some motifs, there is often as few as only two conserved amino acids (for example, the ELM class for the PCSK cleavage site CLV_PCSK_PC1ET2_1). As there are 20 amino acids in total, this limits the sought-for SLiM space to 400 combinations, which corresponds to the pattern probability of 0.0025. This means that for a protein of approximately 400 amino acids length, the expected e-value to find the SLiM just by chance will be ~1. Note: given the background amino acid frequencies, the e-value for the hypothetical SLiM `WW` will be somewhat lower, while for the SLiM `AA` it will be somewhat higher. Moreover, some longer SLiMs have even higher pattern probabilities due to high ambiguity, such as the class TRG_ER_diArg_1, which has the pattern probability 0.0054. The approximate formula that connects the pattern probability ***PP*** and the e-value ***e*** for a pattern of length $\boldsymbol{N_{pat}}$ in a protein of length $\boldsymbol{N_{prot}}$ is (adopted from equation 14 in (Altschul et al., 2001)):

$$e = -(N_{prot} - N_{pat} + 1) * \ln(1 - PP)$$

High pattern probabilities and e-values mean that, in absence of other information, such SLiMs can only be identified at the cost of including a tremendous amount of false positives. To avoid this, there are several approaches possible:

- Comparing the actual pattern probability with that of the background or negative set. This approach is plausible, as some patterns in certain protein groups occur less frequently than they are expected just by background frequencies of individual amino acids. As a result, even 'normal' occurrences may indicate a significant enrichment in this case. Discriminative methods employ this approach.

- Including additional information (e.g., information from evolutionary conservation and flanking amino acids). This approach will effectively lower the e-values, as more conditions need to be satisfied for positive identification.
- Discarding poorly conserved SLiM candidates.

A good *de novo* SLiM predictor employs some combination of the last two approaches. Ideally, the behavior is also tunable, so that the user can select, which filters to apply and which thresholds to use. For example, SLiMFinder uses very strict parameters by default, which lead to high precision but low recall; however, the parameters can be tuned to either increase recall at the cost of precision or further increase precision at the cost of additional losses in recall. Here I describe in brief the statistical model of SLiMFinder, *SLiMChance*.

SLiMChance starts with calculating the probability of $\boldsymbol{L}$ positions forming a SLiM of length $\boldsymbol{L}$ (not including wildcard positions) with $\boldsymbol{d_i}$ different amino acids possible at position $\boldsymbol{i}$ on the basis of background amino frequencies $\boldsymbol{f_a}$:

$$p_{SLiM} = \prod_{i=1}^{L} \sum_{a=1}^{d_i} f_a$$

Then the number of all possible SLiM occurrences in each input protein cluster is calculated. For this, the SLiM candidate is broken into dimers of adjacent non-wildcard positions and the corresponding linkers are retained. For example, a motif `A-x-x-C-[DE]-x-[ILV]` will form the dimers `A-x-x-C`, `C-[DE]`, and `[DE]-x-[ILV]`. For each dimer $\boldsymbol{i}$, the number of its occurrences $\boldsymbol{N_{di}}$ in the protein cluster is calculated. The total number of positions in the cluster, where a dimer of this length can be harbored, is marked as $\boldsymbol{N_t}$. These two numbers are calculated only for unmasked regions of proteins. The possible effective number of SLiM occurrences is calculated on the basis of these numbers and the effective number $\boldsymbol{M}$ of unrelated sequences in the cluster according to the formula:

$$N_m = N_t M \prod_{i=1}^{L-1} \frac{N_{di}}{N_t}$$

Note: ***M*** was introduced in the SLIMDisc algorithm (Davey et al., 2006) under the name 'MST correction'. The probability of the SLiM candidate to occur at least once in the protein cluster is then calculated using the binomial formula:

$$p_c = 1 - (1 - p_{SLiM})^{N_m}$$

This formula can be also viewed as Šidák multiple testing correction (Wright, 1992).

Next, this probability is averaged across all protein clusters and the resulting average is converted to the p-value using the cumulative binomial distribution formula. Finally, this p-value is adjusted, again using the multiple testing correction, for the space of regexes that could have been potentially generated.

The SLiMFinder approach also demonstrates an important difference from classical alignment methods: in a classical local alignment such as is produced for instance by BLAST, all positions are scored, regardless of their conservation. Moreover, a position score must be negative on average for a local alignment algorithm to function properly (Altschul et al., 2001). This means that a longer SLiM with non-conserved positions will score lower than a shorter SLiM with all positions conserved, given that the total number of conserved positions is the same. In case of SLiMs, however, the presence of non-conserved central positions does not mean a lower quality. This is a corollary of the fact that SLiMs are generally a result of a convergent evolution (Chemes et al., 2012). Moreover, to be functional, they need to provide only a few essential, interacting amino acids, which must not necessarily be adjacent. Indeed, some motifs (e.g., the ELM class TRG_NLS_Bipartite_1) show as many as 15 non-conserved positions between the key residues. Thus, classical local alignment approaches are not suitable for SLiM alignments and, consequently, for SLiM search. The same logic applies also for global alignments. Instead, a SLiM search requires an approach, which would not introduce penalties for non-conserved, central columns.

1.8.2. Conserved domains and larger homology regions

As computational SLiM discovery is mainly a problem of finding similar short stretches in protein sequences, the presence of other sequence regions that exhibit similarity is an obvious obstacle and potential source of numerous false positives. The most common 'SLiM competitors' in this sense are conserved domains and larger homologous regions. Conserved domains are longer sequence elements with dedicated functions and a compact 3D structure. A conserved domain functions and evolves independently and can often fold by itself. Conserved domains in different proteins typically have a common evolutionary origin (Liu and Nash, 2012), with some exceptions, where a similar fold or function can arise by convergent evolution (Hudson and Cooley, 2013). Homologous regions represent a common ancestral origin. Next to sequence similarity, homologous regions often show also common functions. Although the underlying biological nature of conserved domains and homologous regions might be quite different, the problem they pose to the SLiM search is essentially the same: they introduce many short, highly similar sequence stretches, which are not SLiMs. In addition, there is an implicit assumption that SLiMs cannot occur inside conserved domains (Neduva et al., 2005). According to this assumption, many SLiM predictors, including SLiMFinder or DILIMOT (Neduva and Russell, 2006), implement conserved domain filters. Nevertheless, no explicit evidence could be found to support this statement. Instead, there are several studies that challenge it. For example, (Mohamed et al., 2015) have shown that different protein- and ligand-interacting interfaces can overlap within the same protein. Moreover, there is a dedicated method for identification of SLiMs that lie specifically inside conserved domains (Horan et al., 2010). In addition, the DILIMOT study itself recognizes the need to turn off the conserved domain filter to be able to recover certain SLiMs, for example, the EB1 motif (Neduva and Russell, 2006).

To deal with homologous regions, one can cluster the input protein set and further consider each cluster as one sequence. Following this approach, SLiMFinder makes corresponding adjustments to score calculations. Although clustering in general sounds like a reasonable approach, it has two major weaknesses. First, it does not provide a good solution for proteins with a homologous region occupying only part (e.g., 50%) of the full-length sequences of the proteins. The percent sequence similarity, as well as the score of the best BLAST hit can be used for thresholding. However, no matter what type

of threshold is used, both decisions are not good: not clustering the proteins together will lead to a quite large homologous region that is retained, while the clustering will remove (or at least discount) quite long unique regions from clustered proteins for further SLiM discovery. Second, as protein similarity measures are not Euclidian distances (e.g., the distance between proteins A and C could be greater than the sum of distances A-B and B-C), forming clusters imposes one more hard choice without a good solution. In the example above, dissimilar proteins A and C would belong to the same cluster, if protein B is similar enough to both of them. This can be the case, if, for example, A and B share a domain D1, while B and C share domain D2. Should then all three proteins be clustered together? On the one hand, if the answer is no, then there would be no practical way to form unbiased and non-overlapping clusters. In this example, the next question would then be, if B should be clustered with A, with C, with both, or with none. On the other hand, if the answer is yes, then huge clusters with up to half of the proteome will form in large data sets (Edwards and Palopoli, 2015).

To deal with conserved domains, one can retrieve annotations for known conserved domains from databases such as SMART (Schultz et al., 1998), CDD (Marchler-Bauer et al., 2015), InterPro (Mitchell et al., 2015) or Pfam (Finn et al., 2016) for further masking. To apply this approach, one should first decide, which database to use and which features to focus on. CDD, for instance annotates structural motifs in addition to conserved domains. In addition, for developing a novel SLiM predictor, one needs to consider dependencies and resulting technical problems, such as access to and updates of the used database(s).

1.8.3. Low complexity regions

Low complexity regions (LCRs) are sequence stretches with strong compositional bias towards a specific amino acid or amino acid group. A popular, more specific definition, which can be easily expressed with an algorithm, defines an LCR as a stretch, in which among each 8 consecutive residues at least 5 are the same. SEG (Wootton and Federhen, 1996) and the SLiMFinder algorithms use exactly these numbers as default parameter values.

As in the case of conserved domains, LCRs are generally considered as incompatible with SLiMs at the same position. Again, no evidence could be found for this statement. On the contrary, some known SLiMs definitely have a low complexity nature. For example, the ELM class LIG_SH3_1 and LIG_EF_ALG2_ABM_2 are P-rich, and LIG_AP_GAE_1 is D/E-rich.

1.9. SLiMs and sequence properties

1.9.1. Surface exposure and SLiMs

As SLiMs mediate interaction of two proteins, they must be accessible to be functional. A SLiM buried deep in a globular protein will in general not be able to bind to other proteins, although there are exception, where the binding site is located in a cleft (Laskowski et al., 1996). Therefore, it makes sense to predict buried regions in order to mask them out for the subsequent SLiM search. The are two measures associated with residue-wise surface exposure: absolute surface area (ASA), measured in square units, and relative surface area (RSA), which is the ratio between the actual ASA and the reference ASA for the same residue in an unfolded, reference peptide (Topham and Smith, 2015). Residues with an RSA below a certain threshold are considered buried.

Although the idea behind the masking of buried regions looks generally sound, there are two possible caveats, in addition to already mentioned cleft binding sites: First, experimentally determined 3D structures are not available for all proteins. As of 28.06.2017, only 441 ELM proteins have a corresponding structure in the PDB database, which corresponds to roughly 20% of all ELM proteins. This means that in most cases, the surface accessibility cannot be calculated directly, but must be predicted. As this prediction is often not perfect, its errors will propagate into the resulting SLiM predictor. Second, a real SLiM may be actually buried in the native protein conformation. It will only become exposed and functional upon induced conformational changes or cleavage (Vallon et al., 2012).

1.9.2. Sequence disorder and SLiMs

Intrinsic sequence disorder defines protein sequence stretches that do not adopt a defined 3D structure. Disordered regions vary in length and are present in at least half of all identified natural proteins (Necci et al., 2016). It was shown that disordered regions

are more likely to harbor SLiMs than ordered ones (Dosztányi et al., 2009). It is speculated that this tendency arises due to the need for a SLiM to adopt a certain conformation in order to bind to its partner. Flexible regions might therefore be preferred, as they can fold into the required structure (Breitenlechner et al., 2004).

Similarly to surface exposure, sequence disorder can be directly inferred only from a 3D structure, or more precisely, the lack thereof. However, in this case the conclusion is not unequivocal, as the failure to crystallize the protein or assess its conformation in another way does not necessarily imply intrinsic disorder. For proteins without a known structure, sequence disorder must be predicted. A popular predictor in SLiM discovery is IUPred (Dosztányi et al., 2005). It shows true positive rate (TPR) of 76.3% and false positive rate (FPR) of 5.3%.

1.10. Evaluating the performance of SLiM predictors

1.10.1. Ambiguity in the performance evaluation

Objective evaluation of the performance is a vital part of the development process of any predictor. It is needed for initial assessment of the method's usability for the given task. It is also needed for assessment of newly added features and parameter optimization. Finally, it is needed for comparison with alternative method to evaluate, if the developed method provides any advantages to the field. Therefore, all the publication of SLiM predictors known to me include at least a short statement on their performance. This also concerns predictors of sequence properties (surface exposure, etc.) mentioned in this thesis.

However, the performance evaluation is not a straightforward, unambiguous task. Developers of different tools use different metrics to evaluate their tools. Furthermore, even if the same metrics are used, they are often calculated under different assumptions and with deviations in formulas. This makes the direct comparison of different predictors quite complicated.

Such differences and deviations do not imply that some authors evaluate their predictors in a wrong way. Instead, the problem of comparing positional data – in this case, positions of SLiMs in sequences – is complex and multi-faceted by itself. In the next

subsections, I introduce the sources of ambiguity that may arise while evaluating the performance of a SLiM predictor.

1.10.2. Resolving overlaps and duplicates of predicted and annotated SLiMs

All the tools tested, excluding MEME (Bailey et al., 2006), return predicted SLiMs that are overlapping or duplicated. In order to deal with these motif instances, some questions must be answered before processing and/or evaluating the results. The main two questions are:

1. Should the predicted motifs be treated separately or merged on overlaps and duplications?
2. Are the residues that are traversed by multiple motifs predicted with more confidence?

In addition, annotated, so-called 'golden-standard' SLiMs in ELM can be overlapping, even within the same class (e.g., LIG_SH3_3 in the protein P25049). To be able to measure the prediction accuracy, one needs to define the approach on how to treat these overlaps.

1.10.3. Assigning matches between predicted and database SLiMs

To evaluate the performance of a SLiM predictor, one needs to compare its results with a benchmark database, such as ELM. The prediction and the database can both be seen as annotations of sequence elements in a given set of proteins. Each record in such an annotation describes one motif site. It consists of a protein identifier, a start and an end residue. The task of the performance evaluation process is to evaluate, how similar two sets of motif sites are.

Ideally, one would like to answer the question, if a particular real SLiM was recovered by the predictor or not and if a particular predicted SLiM matches a real one or not. A good illustration is the study describing NestedMICA (Doğruel et al., 2008), where each database motif is classified as either 'correctly recovered' or 'missed'.

However, the situation is not always clear-cut. Sometimes, a predictor returns a site that overlaps only to a small extent with a database SLiM. The overlap can be as little as one

residue. Furthermore, a predicted site can be much longer than the real SLiM it completely recovers. Therefore, a predictor might be trained to return only very long sites that are able to cover enough real SLiMs merely by chance. If the database and/or the prediction contain multiple, closely spaced or even overlapping sites in the same protein, unambiguous matching and subsequent performance measuring become even more challenging.

Thus, as part of the development process of a SLiM predictor, one should also carefully design the motif matching scheme to be able to train, evaluate, and compare the predictor. The problem was outlined in (Song and Gu, 2015); however, the proposed solution still did not consider all cases of ambiguity.

1.10.4. Counting true negatives in SLiM prediction

The task of a SLiM predictor is not only to correctly identify real SLiMs, but also to avoid making false positive predictions. This means that it must also correctly identify non-SLiM regions as such. Intuitively, one could say that the longer the protein sequence in comparison to the SLiMs is, the more challenging this task becomes. The direct measure for evaluating the correct identification of negative elements is specificity – the share of negative (i.e., absent in the database) elements that were also predicted as negative (i.e., absent in the prediction).

This calculation is straightforward, when the element to count is a single residue, as the number of negative residues is exactly known. However, the corresponding site-wise calculation is ambiguous, as there is no strict definition of a non-SLiM protein site. Should one count all combinatorially possible peptides at all possible starting positions and of all possible lengths that do not contain any SLiM residues as negative?

1.10.5. Choosing the main statistical metric for SLiM prediction

There are several statistical routines that can be used to evaluate the performance of a positional data predictor.

First, it is necessary to calculate similarity scores. These may simply represent a share of correctly predicted elements within some category. For example, the recall is the share

of true positives within true elements. Other simple measures of this type are precision, specificity, and accuracy. Alternatively, similarity scores can combine several simple measures. For instance, the F1 score is the harmonic mean of recall and precision. One can also consider element counts in a more comprehensive manner. The Matthew's correlation coefficient would be an example of this measure. The similarity scores, being just numbers, provide the direct answer to the question of how good the predictor performs from a certain point of view. They always relate to a specific configuration of the predictor's input settings; however, they can be plotted against values of the input parameters to assess the predictor's robustness. An example of a thorough application of similarity scores for evaluating different SLiM predictors is provided in (Song and Gu, 2015).

Second, one can make use of ROC curves (Beck and Shultz, 1986). This approach consists in measuring the true positive rate (TPR) at different levels of false positive rates (FPR). This method is by design more robust, as it assess the whole range of performance, and not just a single point. The user can choose from it the most suitable configuration on the basis of a maximal tolerable FPR and/or a minimal tolerable TPR. Area under the curve (AUC) (Hanley and McNeil, 1982) represents another number that can be used as a performance measure. Although being very illustrative and robust, ROC curves cannot be built for all types of predictors. One of the necessary conditions is the performance dependence on the parameter of a single tunable real number. Another condition is the absence of local maxima and minima in the functions of TPR and FPR on this parameter. This means that with an increase in the parameter, one should not observe a positive change in TPR or FPR coupled with a negative change in FPR or TPR, respectively. Situations, when TPR or FPR rise in one range of parameter values and falls in another range are not acceptable. If these conditions are not met, there will be potentially several different values of TPR or FPR for a single given value of FPR or TPR, respectively, which will break the concept of the ROC curve. In some cases, this effect can be addressed by building multiple curves. For example, if there are multiple input parameters, a curve can be plotted for each parameter separately, with all the others being 'frozen'. ROC curves were used to demonstrate the performance of `hhsearch` in homology searching (Söding, 2005), as well as some SLiM predictors (QSLiMFinder (Palopoli et al., 2015)).

Third, one can perform a statistical enrichment analysis with significance score(s) as output. The most useful type of such an analysis is the hypergeometric analysis (Taskesen et al., 2013) implemented, for example, in the software tool diffReps (Shen et al., 2013) to evaluate and compare ChIP-Seq data.

While building a performance report on a given predictor or an overview of several predictors, one can compute multiple measures of different types for the purpose of comprehensiveness. However, while developing and optimizing a predictor, one must focus on a single measure. The procedure can be also more complex, taking into account many separate measures, for example, through averaging. Not all abovementioned measures are suitable to be used as such a performance metric. For example, if the recall is chosen as the main metric, a predictor that always classifies all input data as positive, will be the winner. If the precision is chosen, the high-performer will be a predictor that positively identifies only the small fraction of positive elements that is easiest to detect. Both these predictors will be practically useless for the user, despite scoring high. Therefore, the resulting metric for optimization should satisfy the following requirement: it reaches its maximal value if and only if the prediction reproduces the benchmark annotation perfectly. If non-perfect predictions can also result in the maximal score, the optimization becomes problematic.

1.11. Visualization of SLiM prediction results

1.11.1. Plain text and pseudographics for SLiM representation

Plain text with a clear pre-defined structure is the most simple program output that can be produced. It has the advantage of also being easily machine-readable. It is universal and does not depend on additional software. That is why all known SLiM predictors can produce plain text output, even if they also support more sophisticated formats. Properly used spacing and special characters (i.e., elements of pseudographics) help to show the SLiM alignments, scores, and other supporting information in a way that is easily understood by humans. MEME (Bailey et al., 2006), for example, draws a simplified position-specific probability matrix for each identified motif. The pseudographics is dependent on the font with the main restriction that the font must be monospace.

1.11.2. Image files

SLiMs can be also represented as images. Sequence logos represent the most popular images for SLiMs (Schneider and Stephens, 1990). The current software solutions to build the logos from the alignments include WebLogo (Crooks et al., 2004) and IceLogo (Colaert et al., 2009). The logos depict the enrichment of the corresponding amino acid types through the height of the letters in each position of the SLiM. The enrichment is by default measured as information content. In addition, the gap frequency may be depicted through the letter width (the more gaps the thinner the letters).

1.11.3. Interactive web-based interfaces

There is quite a lot of information associated with SLiMs. This includes SLiM instances themselves, flanking residues, positions in whole protein sequences, conservation, scores, etc.. It is therefore not easy to display it all at once without overloading the visual field of the user. A good solution for this problem will make use of both, text properties (font, case, size, weight, color, etc.) and graphical elements (images, lines, arrows, etc.) to encode as much information about the SLiM as possible in an intuitive manner. In addition, interactiveness can be used to improve the user's experience.

1.12. Relation to published and submitted manuscripts

The ideas and results of this thesis has resulted in one published and one submitted manuscript. HH-MOTiF was published as (Prytuliak et al., 2017). SLALOM was submitted as "SLALOM, a flexible method for the identification and statistical analysis of overlapping continuous sequence elements in sequence- and time-series data" to *BMC Bioinformatics*. My contribution in the first manuscript was designing the algorithms, developing the software, conducting performance tests, and assist in writing the text. The contribution of co-authors was as follows: Michael Volkmer – technical support, Markus Meier – implementing required modifications to HH-suite, Bianca Habermann – supervising the project, correcting and writing the text. My contribution to the second manuscript was designing and developing the software, analyzing data and assist in writing the text. The contribution of co-authors was following: Friedhelm Pfeiffer – providing and pre-processing the genome data, assist in writing the text, Bianca Habermann – supervising the project, writing the text.

The first manuscript describes the web-tool HH-MOTiF and represent an intermediate state of the thesis project. The thesis goes deeper in detail concerning the method descriptions. In addition, it focuses more on the question of how the ideas and algorithms behind HH-MOTiF can contribute to other applications as well as formulates and develops further the hypothesis on impact of short motif diversity on the accuracy of their computational recovery.

The second manuscript in its current state describes the standalone tool SLALOM. It includes detailed explanations of all the implemented operation modes together with the user instructions, which remained out of the scope of this thesis. On the other hand, the thesis provides more extended examples ('case studies') to demonstrate the added value of SLALOM.

2. METHODS

2.1. Development procedures and techniques

2.1.1. Programming languages, libraries, and auxiliary tools

The main back-end application of HH-MOTiF is written in the Python programming language. The syntax is compatible with version 3.4 or higher. Besides the standard modules, the following libraries are used in HH-MOTiF:

- `numpy`: for compact data storing
- `sharedmem`: for sharing a `numpy` array between different processes
- `mpi4py`: for implementing MPI enabling running HH-MOTiF on a cluster

The web-server wrapper application operates within the Django framework with MySQL as the broker. The corresponding additional Python modules (with dependencies) are used:

- mysql.connector
- django

The plots were prepared with R in the IDE program RStudio. The package `ggplot2` was used for plotting with the additional packages `plyr` and `reshape` for data preparation as well `RColorBrewer` for better visualization.

The web pages of the server are served in HTML5. The Bootstrap library of version 3.3.7 was used to design the site with CSS3 and JavaScript elements. The JQuery library of version 3.3.1 was in addition used for the flow control scripts.

The C++ programming language of the standard C++14 was used to generate the GI lists for separate taxonomic units.

2.1.2. Performance tests

Three types were conducted to boost the performance of HH-MOTiF as well as to prevent running application crashes:

- Runtime comparisons of different implementations of the same mini-tasks in Python. The mini-tasks corresponded to calculation types being performed by HH-MOTiF. The implementations compared:
    - generic programming vs. OOP
    - Python objects vs. `numpy` arrays
    - `multiprocessing` vs. MPI
- Wall clock runtime measures of the running HH-MOTiF back-end application. These are done with the `time` module from the main process.
- Per-process peak RAM consumption measures of the running HH-MOTiF back-end application. The data are read from the Linux system file '`/proc/{pid}/status`'.

2.2. HH-MOTiF pipeline

2.2.1. Pipeline overview

After checking the input for validity, HH-MOTiF in the *de novo* mode starts processing of the query sequences by searching the orthologs. The retrieved orthologs are aligned and converted to HMM profiles. In parallel, sequence masking is performed according to the user-defined criteria. Then, the masked HMM profiles are compared to each other in an all-to-all, pairwise manner. The best alignment hits identified in each pair are integrated into hierarchical structures, which are termed 'motif trees'. Next, the generated motif trees are evaluated by several additional criteria and, if needed, trimmed or even completely discarded. The retained trees are shown to the user on the interactive web page.

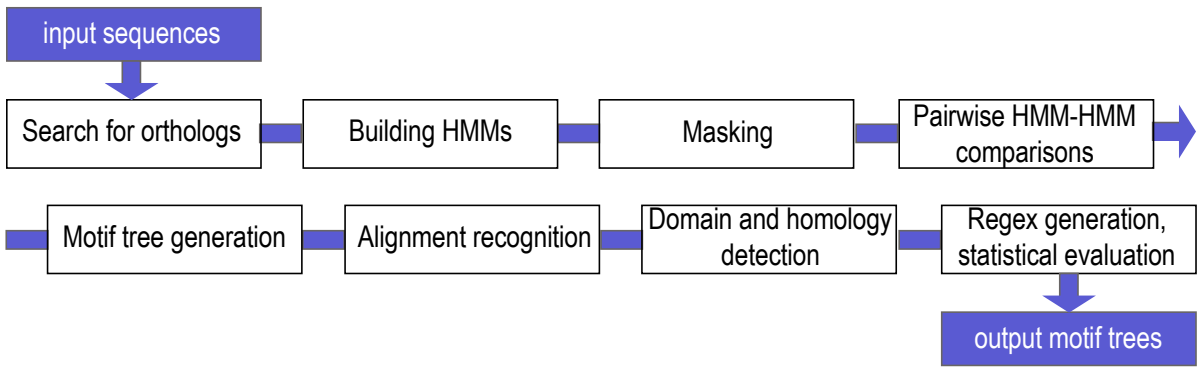


Figure 1. The separate steps in the HH_MOTiF pipeline as they are implemented in the backend application

The pipeline is schematically shown in Figure 1.

2.2.2. Search for orthologs

The utility `blastp` from the NCBI BLAST+ standalone package is used to find the BLAST hits (Altschul et al., 1990). The search for orthologs is performed as the reciprocal best BLAST hit (Bork et al., 1998) detection.

The initial BLAST is performed against the `nr` database for each query sequence and saved in a separate file `input_temp.fasta`:

```
blastp -db nr -query input_temp.fasta -evalue 1e-10 -
num_alignments 2000 -outfmt 5 -out blast_temp.xml -num_threads
32
```

The generated file `blast_temp.xml` is parsed to read the information on the BLAST hits.

In this file, the top hits, the query itself is searched for. All hits belonging to the same species as the top hit with 100% identity and length in range [L, L+2], where L is the length of the query, are considered as the 'self-candidates'. If no self-candidates can be identified (i.e., there is no hit displaying 100% identity and length in the required range), the ortholog search is interrupted. Otherwise, the species of the top qualified hit is resolved to the NCBI taxonomy identifier according to the file `gi_taxid_prot.dmp` (downloaded from the NCBI FTP server). The files containing GI lists for each species are pre-generated with a custom C++ program.

Then, the best BLAST hit for each species with coverage of at least 90% and identity of at least 70% but at most 95% is back-BLASTed against the species of the query (i.e., of the top qualified hit of the initial BLAST). If this species belongs to the smaller database `refseq_protein`, it is used instead of `nr`. A separate FASTA file is generated for each hit being tested:

```
blastp -db refseq_protein -query hit_{hit_id}.fasta -evalue
1e-10 -num_alignments 2000 -outfmt 5 -out blast_temp.xml -
num_threads 32 –gilist gi_{tax_id}.txt
```

Pre-generated GI lists and sub-database identification do not influence the results but enhance the performance many-fold. As `blastp` is not capable of efficient parallelization of searches against single species, the reciprocal BLASTs are themselves performed in parallel in a pool of processes.

If the top hit in the back-BLAST search is one of the self-candidates, the hit is confirmed as an ortholog; otherwise, it is rejected.

The final list of orthologs is saved as a non-aligned FASTA file.

2.2.3. Building hidden Markov models

The FASTA file with orthologs is aligned with MAFFT v. 7 (Katoh and Standley, 2013) to produce the global MSA:

```
mafft --maxiterate 1000 --thread 32 --globalpair orthologs.fasta > orthologs_aligned.fasta
```

The alignment file is then subjected to the utility `hhmake` from HH-suite (Söding, 2005) (Remmert et al., 2011) to generate the corresponding HMM. The option '`-M first`' forces using of the positions of the query sequence as match states regardless of their conservation (i.e., the resulting HMM is 'centered' around the query), while the option '`-id 100`' prohibits removing any of the identified orthologs:

```
hhmake -i orthologs_aligned.fasta -o query.hhm -M first -id 100
```

For performance reasons, the `*.hhm` files for all the ELM proteins were cached and not re-calculated during the optimization, unless changes at or prior to the HMM generation step were introduced.

2.2.4. Masking

The query sequences can be masked according to one of three criteria:

1. User-specified masking file.

2. Surface exposure.
3. Sequence disorder.

The user-specified file contains the regions (start and end residues) of interest for each input protein. All the residues that do not belong to these regions, are masked. Query proteins not mentioned in this file remain unmasked.

Surface exposure masking is performed through predicting the surface exposure with NetSurfP (Petersen et al., 2009). The input file is re-used from the ortholog search procedure:

```
netsurfp –i input_temp.fasta > netsurfp_output.txt
```

The NetSurfP output contains residue-wise RSA values. The residues, whose value is below a certain threshold (default value: 0.16), are masked.

Sequence disorder masking is performed through predicting residue-wise disorder with IUPred (Dosztányi et al., 2005). All residues with an IUPred value below 0.2 are masked. The IUPred program is run with the option '`short`':

```
iupred input_temp.fasta short > iupred_output.txt
```

The user can select the three masking approaches independently. If several of them are applied, the masks are merged, so that a residue that is masked in at least one of them is considered masked.

The resulting mask is resolved so that there are no regions less than 3 residues with different masking status (either masked or unmasked). To resolve the termini, the sequences are viewed as being flanked by infinitely long masked regions. To achieve this, the following algorithm is implemented:

1. Determine all the regions that already satisfy the length requirement and keep them unchanged.

2. For remaining regions, give them the masking status of the majority of residues within them, and rendering them non-masked, if the number of masked and unmasked regions are equal.

The final mask is saved as the text string listing the regions of masked residues separated by commas. A region is represented as the start and end residue numbers separated by a dash. Lone terminal residues are listed as single numbers. If the sequence is completely unmasked, the string is empty. This format is compatible with the HH-suite utilities `hhalign` and `hhsearch`.

2.2.5. Pairwise HMM-HMM comparisons

After the HMMs from the query sequences are generated, they are compared with each other in a pairwise, comprehensive manner. To this end, the utility `hhalign` from the version 3 of HH-suite is used. It accepts two HMMs and two corresponding masks as input and produces the list of alignment hits placed in an `*.hhr` file as the output. The 'input' is submitted with the option key '`-i`' and the 'template' with the option key '`-t`'; flipping the input files does not influence the results. Masks are submitted by the options '`-excl`' and '`-template_excl`' respectively. An example of `hhalign` command line call is provided below:

```
hhalign –i query_A.hhm -t query_B.hhm -o output.hhr -smin 0 -
alt 100 -gaph 999.0 -gapi 999.0 -norealign -nocontxt -excl
1,9-33,42-145,286-312 -template_excl 118-214,342-395
```

The options '`-smin 0 -alt 100`' force `hhalign` to output at least 100 hits to choose from. Although only few hits will be selected to proceed to the next stage, this is needed in case the top hits are not suitable according to the additional criteria.

The options '`-gaph 999.0 -gapi 999.0`' are applied, when the gap restriction is chosen by the user. The gap restriction will not allow gaps longer than 1 residue in the SLiM alignments. The options achieve this by effectively setting the gap continuation penalty to infinity.

The options '`-norealign`' and '`-nocontxt`' switch off the maximal accuracy (MAC) algorithm and the context term to the score, respectively. The user of the web application does not have control over these options. Nevertheless, they were tested during the optimization of the back-end application.

The alignment hits in the output are sorted by their Viterbi scores. The hits not satisfying the following criteria are not considered:

- The number of aligned columns is in the range [3, 30]
- The Viterbi score is in the range [11.0, 40.0]

The number of retained hits, as well as the number of columns and Viterbi score ranges cannot be changed by the user of the web application. However, different values were tested during the optimization process. From the retained hits, the four with the highest Viterbi scores are selected.

An alignment hit in the output file is represented as alignment with some additional information (an example is provided in Section 1.7.3. HH-suite). The start and end positions, the Viterbi score, the two gapped sequence stretches, and the alignment signs (the middle line) for the retained alignment hits are saved in a structured `numpy` array.

2.2.6. Motif tree generation

The retained alignment hits are accumulated to detect the most plausible SLiM locations. For each query in an input set of $\boldsymbol{N}$ sequences, two distinct values are calculated residue-wise: the average Viterbi score of all the hits traversing this position and the number of other query proteins these hits involve. The latter can vary in the range [0, $\boldsymbol{N}$-1]. If it is positive, the former can vary in the range [11.0, 40.0], which is same as the score of each individual hit. The residues that have hits with less than $\boldsymbol{N_{min}}$-1 other queries, as well as residues with the average Viterbi score below 13.0, are masked from further consideration. $\boldsymbol{N_{min}}$ is a function of $\boldsymbol{N}$ (for details, see Section 2.4.6. Statistical model for false positive prediction). The unmasked residues form putative SLiM regions. Regions shorter than four residues are discarded. The score threshold and the minimal length of the region cannot be changed in the web application; however, different values were tested during the optimization. Each retained region becomes a so-called 'motif root'.

The corresponding aligned sequence stretches in other queries from the associated hits become 'motif leaves'. Although a motif leaf is usually shorter than the original sequence stretch in the hit as it represents only the part that is aligned to the motif root, its score is left unchanged. A motif root with corresponding motif leaves form a 'motif tree'.

2.2.7. Alignment recognition: the algorithm for motif tree trimming

As a motif tree represents a true MSA with only root-leaf but not leaf-leaf pairs being aligned, it may turn out that the leaves do not show enough similarity with each other to form a putative SLiM. Therefore, a technique called 'alignment recognition' was developed to trim and, if needed, to completely discard an obtained motif tree.

A putative SLiM alignment may not be suitable in two distinct ways:

1. Motif leaves are aligned to different parts of the motif root. In this case, the root is quite long, much longer than an average leaf, so that many leaf pairs do not even overlap with each other (see leaves C and D in Figure 2).
2. Motif leaves are aligned to the same part of the root, but show strongest matches in different positions. In this case, there are too few alignment columns with good matches in the majority of the leaves. An extreme case would be a situation, where certain leaves show strong matches in odd columns but mismatches in even columns, and vice versa for the other leaves (see leaves A and B in Figure 2).

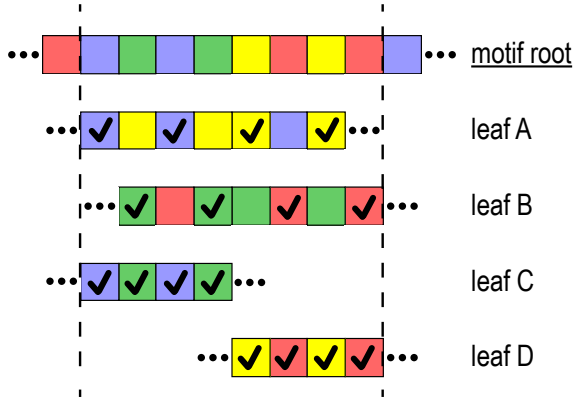


Figure 2. Schematic example of a motif tree in which all the leaves are aligned well with the root but show nevertheless poor consistency with each other. Squares of different colors depict residues of different types. Those with a tick match to the corresponding residue in the motif root. The dashed vertical lines mark the borders of the motif tree. Although leaves A and B match the root in 4 positions each, they do not have a single match between each other. The same is true for the pair C and D.

In both cases, a situation arises, in which it is not possible to summarize the SLiM with a reasonable regex that would describe the majority of the leaves. Such motif trees are unlikely to represent instances of real SLiMs.

There may be two underlying reasons for bad alignment quality of a motif tree. First, the motif tree can arise completely by chance from non-related leaves. Second, the motif tree can represent a real SLiM, but be contaminated by a few, non-related leaves displaying similarity to the root only by chance.

The alignment recognition algorithm should be able to selectively prune the tree in the latter case and completely discard it in the first case. The proposed realization of the algorithm attempts to do so by applying an iterative trimming procedure. The trimming can proceed both, in horizontal direction, shortening the motif root and corresponding leaves by discarding columns at the edges; and in vertical direction, discarding distinct leaves showing the least similarity with the rest of the tree. The quality of a match in individual root-leaf pairs is determined on the basis of the alignment signs from the `hhalign` output: '`|`' and '+' mean a good match and are assigned 2 points as the position score; '**.**' means a mediocre match and is assigned 1 point; '−' and '=' mean mismatch and are assigned 0 points. The algorithm can be outlined as follows:

1. Classify all motif tree columns by the conservation. A column that has 2 points with leaves in at least $\boldsymbol{N_{min}}$-1 query sequences is classified as well-conserved and gets 2 points column-wise. $\boldsymbol{N_{min}}$ is calculated on the basis of the input size $\boldsymbol{N}$ (for details see Section 2.4.6. Statistical model for false positive prediction). A column that fails to be classified as well-conserved, but has at least 1 point with leaves in at least $\boldsymbol{N_{min}}$-1 query proteins is classified as mediocre-conserved and gets 1 point column-wise. All other columns are classified as non-conserved and get 0 points.
2. If the sum of all column-wise scores is less than 6, discard the motif tree and terminate.
3. Discard all non-conserved columns at the borders of the motif tree, so that it begins and ends with either well- or mediocre-conserved column (horizontal trimming).

4. For positional scores of individual root-leaf pairs, reduce these to the column-wise score, if the latter is lower; this step helps to disregard strong individual matches in non-conserved columns.
5. For each individual root-leaf pair, if the sum of newly adjusted positional scores is less than 6, discard the leaf (vertical trimming).
6. If the retained leaves are in less than N_{min}-1 query proteins, discard the motif tree and terminate.
7. If no trimming occurred at the steps 3 and 5, return the resulting tree and terminate.
8. Go to the step 1.

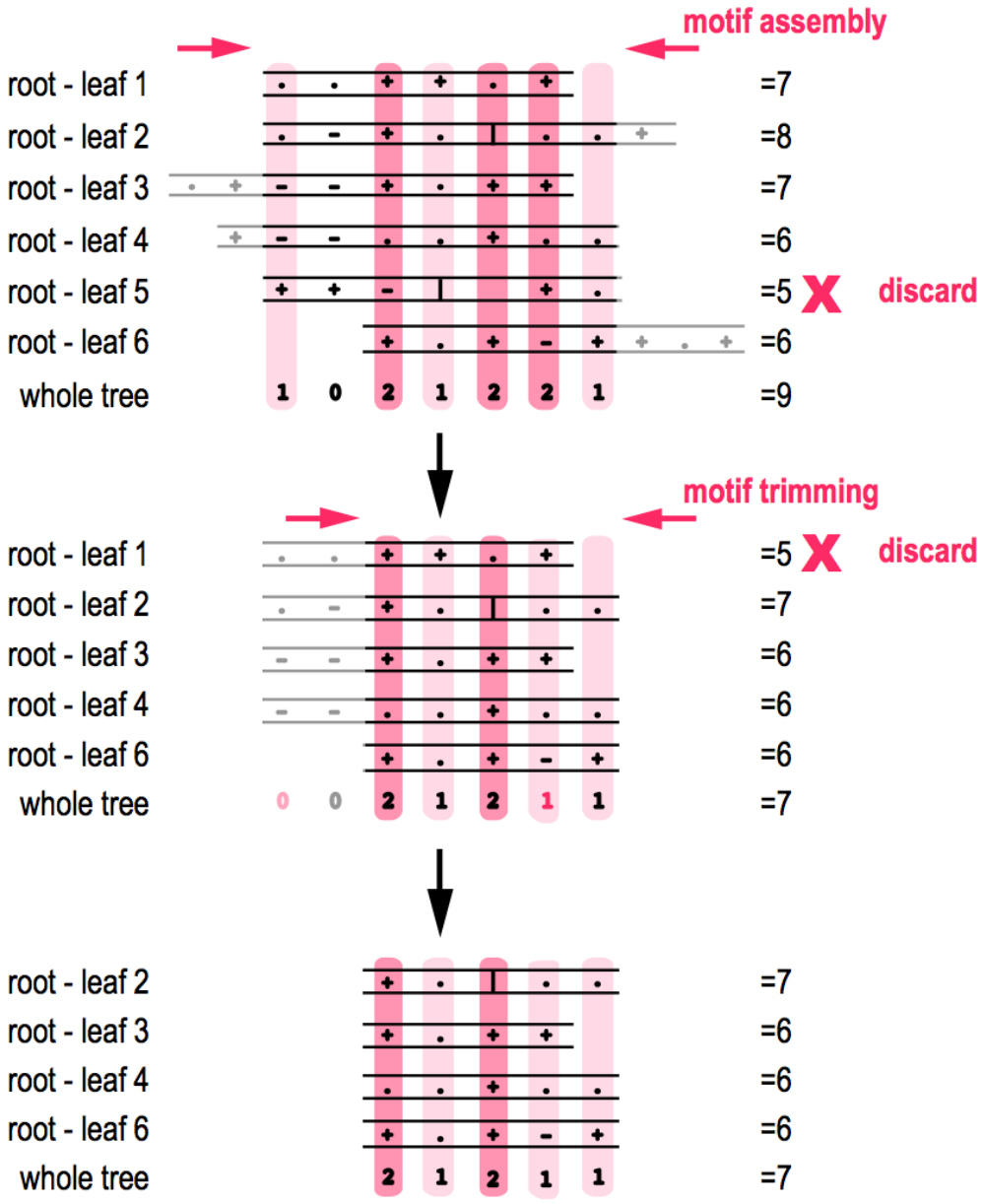


Figure 3. Schematic example illustrating the alignment recognition algorithm. All leaves are located in different query proteins. The tree must have at least 3 leaves (i.e., N_{min} being 4). The alignment signs ('|', '+', '.', '–', and '=') are produced with `hhalign` and represent column-wise quality in pairwise root-leaf alignments. The signs are scored as follows: 0 ('=', '–'), 1 ('.') or 2 ('+', '|'). The scores are first assessed vertically to produce column scores. The column score is 2, if there are at least N_{min}-1=3 root-leaf pairs with a minimum score of 2; the column score is 1 when the condition for scoring 2 is not satisfied but there is at least 3 root-leaf pairs scoring at least 1 in it; the column score is 0 otherwise. Dark red columns are highly conserved (scoring 2), while light red ones are moderately conserved (scoring 1); white columns are non-conserved (scoring 0). After the column scores are calculated, the maxima of the actual alignment sign scores and the corresponding column scores are summed up horizontally to produce leaf scores. In this way, a leaf score cannot exceed the whole tree score, which is the sum of the column scores. Leaves with a score less than 6 are discarded and the procedure is repeated iteratively until conversion. Column scores that have changed are shown in red. The figure was reprinted from (Prytuliak et al., 2017) in agreement with the license no. 4200740316935. The copyright belongs to Oxford University Press.

The algorithm is also illustrated in Figure 3.

Finally, in the retained, trimmed motif trees, sums of positional scores of leaves in the same query protein are compared and leaves with lower sums are discarded (note that discarding suboptimal leaves in the same query protein does not trigger recalculation of the whole tree, as N, N_{min}, etc. refer to the number of *proteins* in which the SLiM candidate occurs and not the number of leaves). If a tree has leaves with the same sum in the same query protein, they are all retained.

2.2.8. Domain and homology detection

A motif tree of high alignment quality may arise not only out of genuine SLiMs but also out of longer homologous regions in query proteins. These regions can reflect overall evolutionary and functional proximity of these proteins or a shared conserved domain. Therefore, HH-MOTiF implements an algorithm to prevent homologous regions from being identified as SLiM candidates.

The HH-MOTiF homology filter, like the primary detection of the SLiM candidates, is based on the Viterbi alignment scores and aligned column counts of alignment hits from the `hhalign` output. However, unlike the SLiM detection, it looks for very long and high-scoring hits, namely the ones that satisfy at least one of the two conditions:

1. The Viterbi score is at least 150.
2. The number of aligned columns is at least 90% of the length of the shortest query in the pair.

These long, high-scoring alignment hits are hereafter referred to as *homology markers*. Each homology marker, therefore, is a pair of aligned sequence stretches from two different query proteins. The motif trees retained after the trimming and filtering steps are checked for their location to homology markers. Ideal SLiM candidates should avoid locating in a homology marker. Only leaf-leaf pairs need to be checked at this point, as root-leaf pairs have already passed a much stricter length and score criteria (see Section 2.2.5. Pairwise HMM-HMM sequence comparisons).

Simple checking itself is performed in the following manner: if two leaves are within the *same* homology marker by at least 3 residues, they are considered homologous. This consideration, however, always relates to a *pair* of leaves in a given motif tree, and not to individual leaves or to individual query proteins. If only one of the two leaves locates to a homology marker, or the leaves locate to different markers, they are considered non-homologous.

After all homologous leave pairs are identified, the total number of query proteins, to which the motif tree locates, $\boldsymbol{N_{tree}}$, is adjusted correspondingly to generate the corrected protein count $\boldsymbol{N_{corr}}$. If no homologous leave pairs are detected, $\boldsymbol{N_{corr}}$ equals $\boldsymbol{N_{tree}}$. Otherwise, for each group of leaves interconnected through homology in $\boldsymbol{N'}$ queries proteins, $\boldsymbol{N_{corr}}$ is reduced by $\boldsymbol{N'}$-1, as if only one leaf from the group were retained. If $\boldsymbol{N_{corr}}$ becomes less than the minimal required number $\boldsymbol{N_{min}}$ of proteins to participate in a SLiM, the whole tree is discarded. Otherwise, all leaves are retained in the output. However, $\boldsymbol{N_{corr}}$ is used instead of $\boldsymbol{N_{tree}}$ for subsequent calculation, as lowering the value negatively impacts the SLiM score.

2.2.9. Regex generation and statistical evaluation

After the trimming and filtering of motif trees on the basis of general alignment quality, the retained trees may still be of insufficient quality to be considered as SLiM alignments. There may be five issues degrading the motif tree quality:

1. Although being conserved in the majority of leaves, a well-conserved column can still have leaves in remaining query proteins with a too strong mismatch.
2. Mediocre-conserved columns can show too much variety in amino acid composition. This is an especially acute case for columns with Asn as the

dominant amino acid in the root, as Asn aligns with a negative score less frequently than other amino acids. Asn aligns with 10 amino acids with a non-negative score in the BLOSUM62 matrix; alignment scores in HH-suite are more complex, but the tendency is the same.

3. The matches are achieved with too frequent amino acids in the input dataset.
4. The residue combinations that make the conserved part of the motif tree are too typical for the query sequences (e.g., a hypothetical motif WDA in a protein with several WD domains).
5. The query proteins are very long and putative SLiMs have a higher probability to occur just by chance.

To check, if these issues are relevant for a specific motif tree, its regex is built. In this regex, the non-conserved columns, which scored 0 at the tree trimming step become wildcards and are displayed as dots, as the Perl syntax is applied. If a column contains at least one gap, it is also treated as non-conserved. For all other columns, all the occurring amino acids are listed – in square brackets, if there are several – including those that are located in the leaves having the positional score 0 in this column. If there are several leaves in a single protein, amino acids from all leaves are considered. Therefore, the rules for regex generation are stricter than the criteria for alignment recognition.

For each generated regex, its p-value – the probability to occur just by chance in the input dataset – is estimated. The lower the regex p-value, the more reliable is the predicted SLiM candidate. The issues 1-3 listed above are addressed by calculating total frequencies $\boldsymbol{f_i}$ of all amino acids occurring in each conserved column $\boldsymbol{i}$ out of total $\boldsymbol{C}$ conserved columns as sums of corresponding background frequencies in the input data set. The issues 4-5 are addressed by calculating the total number $\boldsymbol{T}$ of combinations to construct each motif element – either root or leaf – from the corresponding 2-residue blocks, hereafter referred to as dimers, available in the respective query sequence. The lower the frequencies and the dimer counts, the lower is the p-value.
The Šidák correction formula is used for calculating the p-value:

$$p = 1 - (1 - \prod_{i=1}^{C} f_i)^T$$

The combination count ***T*** is calculated as the product of ***C***-1 dimer counts along the ***C*** conserved columns of the regex. A dimer is a regex that includes exactly two neighboring conserved positions of the motif tree regex, including the linker (wildcard position) of exactly the same length. For example, the hypothetical motif tree regex `[ILV]..[DE]C` forms two dimers: `[ILV]..[DE]` and `[DE]C`. To get the count for the first of them, all occurrences of `I..D`, `I..E`, `L..D`, `L..E`, `V..D`, and `V..E` will be counted and summed up in each of the query sequences, the motif tree locates to. Similarly, the occurrences of `DC` and `EC` will be counted and summed up. As the sequences do contain the motif, at least one dimer for each column pair will be counted. If the regex dimers are too typical for the current query protein, ***T*** can become very large, indicating low significance of the motif tree.

The displayed regex p-value is the ***p*** averaged through all the retained leaves of the motif tree. It should be noted that it represents the p-value of a *regex,* not the underlying motif tree, as the conversion of a tree to the regex loses much information about the potential SLiM candidate. Therefore, a quite high threshold value (0.3) is used by default. However, this threshold can be changed by the user.

2.2.10. SLiM visualization

After the SLiM prediction is finished, the user can access the HTML page with the results (see example on Figure 4). The page includes the FASTA-formatted proteins from the input dataset. It preserves the original headers or the filenames (depending on the submission mode) as well as the sequences themselves, 80 residues per line with residue numbers at the end of each line.

At the top of the page, there is information on the total number of identified motif trees followed by the list of the motif roots sorted by the position in the input dataset. Upon clicking on a listed motif, the page is scrolled to the position of the motif root in the dataset and the motif tree gets selected. By default, the motif roots are presented in red and underlined in the dataset. Upon its selection, the color changes to purple, the root gets enlarged, and the associated motif leaves appear in purple. In addition, dashed lines connect the root with the leaves. If a leaf overlaps with another motif root, the

overlapping stretch changes the color to purple but remains underlined, so that such overlaps can be easily seen.

After selecting a motif tree, a panel appears on the right side. This panel contains the sequence logo (generated with WebLogo (Crooks et al., 2004)), as well as the information box with the regex, the regex p-value, the averaged alignment score, and the pseudo-MSA. In addition, there is a link to download the pseudo-MSA as a FASTA file. This file can be directly submitted as input motif in the proteome-wide search of HH-MOTiF.

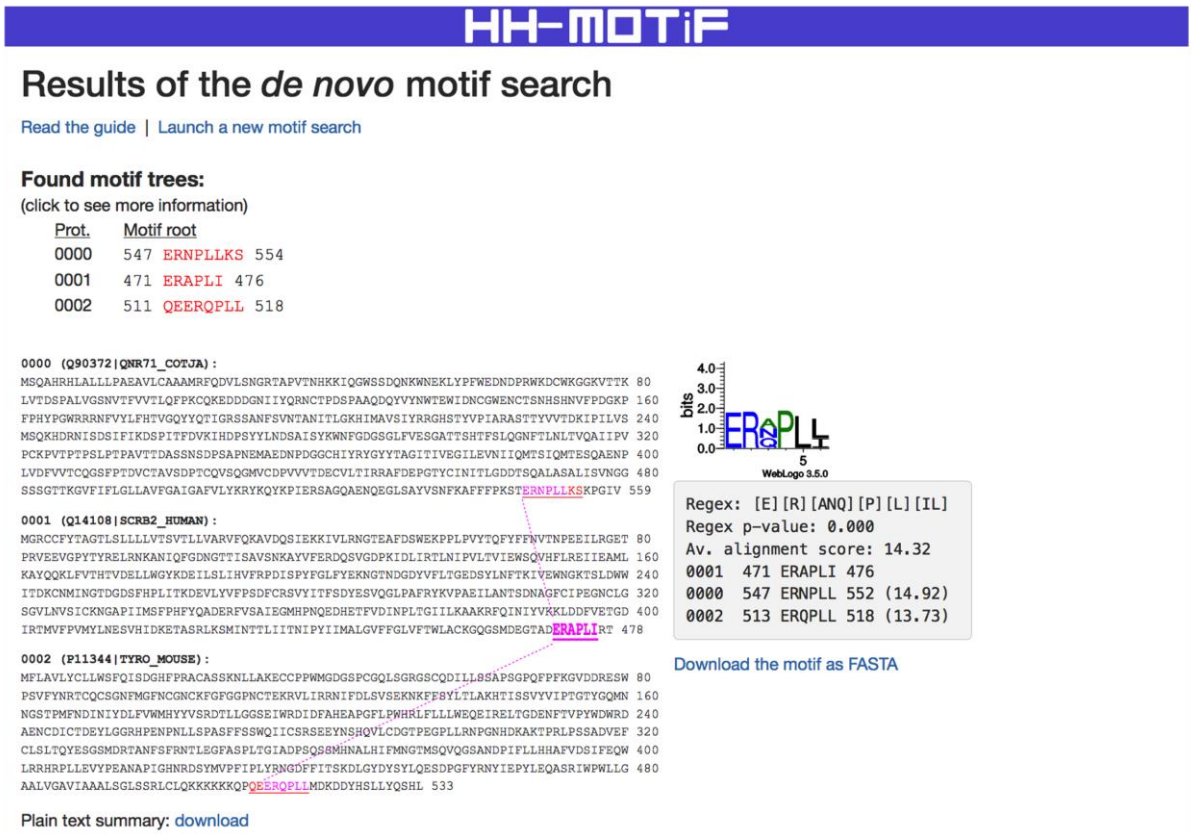


Figure 4. Example of the HH-MOTiF output web page for a *de novo* motif search. Three motif trees are identified in this search. Their roots are underlined. The motif root of the selected tree is shown in underlined bold purple with the associated motif leaves displayed in plain purple; the root is connected to the leaves with dashed lines. The sequence logo, the (pseudo-)MSA, the alignment scores and the regex are shown at the right. The figure was reprinted from (Prytuliak et al., 2017) in agreement with the license no. 4200740316935. The copyright belongs to Oxford University Press.

The sequences are placed inside HTML `span` tags. Upon changing of color or style, a new object is started. The motif roots are stretched vertically upon selecting with the `scale` property. The FASTA headers are coded as boxes (in `div` tags) to limit their

width to the width of sequences. The dotted lines are coded as boxes using the `rotate` transformation. The information box has the maximal width of 31`em`. Both the logo, in SVG format (an `img` tag), and the pseudo-MSA (in a pre tag) have defined maximal height (35% of the page) and width (30`em`). The pseudo-MSA as text of fixed font size can potentially overflow both, the width and the height. To assure the correct appearance of the sliding bars, the CSS properties '`overflow-x: hidden`' and '`overflow-y: auto`' are used, which prevent the appearance of the vertical bar, which has its own width from triggering the appearance of an unnecessary horizontal bar. To prevent the vertical bar covering the right-most characters of the pseudo-MSA, an extra space is added after each line. The position of the information box is automatically adjusted by the scrolling event listener.

The result page is non-scalable, as zooming distorts the positions of the dotted lines in some browsers. Instead it retains its complete functionality on mobile browsers, although both horizontal and vertical scrolling must be employed.

2.3. Web-server layout

The HH-MOTiF web-server consists of five pages accessible through a menu on top, which is stylized with the Bootstrap library. The links are relative, so that the server can operate as the site root as well as a sub-page. The CSS and JavaScript functionality used support all modern desktop and mobile browsers, starting with IE9. The site does not use cookies.

The main page displays by default the message on disabled JavaScript. If JavaScript is turned on, however, this message is replaced with the actual content – the two levels of Bootstrap tabs enabling switching between three distinct input forms. The outer level – the switcher between *de novo* and proteome-wide searches – is located to the left side on default-sized pages. It is connected with the input forms by one horizontal and one diagonal line. The latter is implemented as a `rotated div` and is adjusted upon form switching. On small screens, the outer level tabs collapse to a hamburger menu. The forms themselves (see Figure 5) are coded as Django templates; the served pages contain the input fields as `input` tags and the submit button as a `button` element. The input is partially checked for validity with JavaScript, so that the user can correct the

errors without communicating with the server. For security reasons, the checks are repeated on the server side, as well.

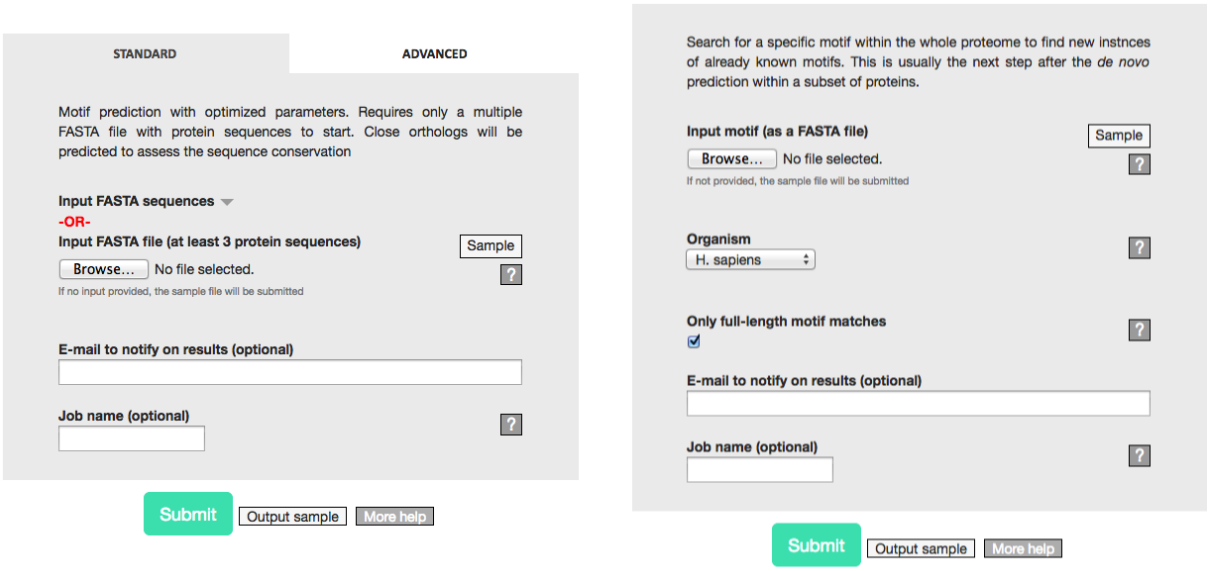


Figure 5. Input forms of the HH-MOTiF web-server: the De novo tab (left) and the *proteome-wide* tab (right). Two modes – **standard** and **advanced** – are supported in the *de novo* tab. In addition to data submission, the forms contain links to the sample input files and output web pages.

After a form submission, the input data are transferred as a POST request to the server for validity checking. AJAX requests with one-second intervals are sent from the client side to get the status. The time out is set to 60 seconds; however, normally, the very first request delivers the status. If the check has failed, the user is returned to the same form with the error message displayed at the bottom. Otherwise, the user receives the link to the job page. The link contains a 40-digit hexadecimal number (job ID) randomly generated on the fly to distinguish between the jobs and to prevent an unauthorized access. For security reasons, this number differs from the initial session key used for AJAX requests.

If the input check on the server succeeds and an AJAX request with the valid session key is received from the client, the server responds with the job ID and proceeds to the actual motif prediction. To prevent rogue requests, the server proceeds only after the AJAX request. There can be up to 10 checking tasks running in parallel but only one single prediction task. The queue can contain up to five waiting tasks. If it is filled, the server stops accepting new tasks.

The communication between the server back-end components is performed through a MySQL database. Separate tables are established for checking and running queues. The worker scripts listen for upcoming tasks to launch the corresponding back-end applications. Another worker launches daily to remove all results more than 7 days old. In addition, a special script is launched each time the input checking is finished. This script will clear the corresponding record from the MySQL table after a 30-second time-out. This prevents database overflows caused by multiple user-interrupted task submissions. MySQL connections themselves are wrapped in pools to ensure stability and to prevent the number of connections from ballooning on reasonably high loads.

2.4. Optimization and evaluation of the HH-MOTiF algorithm

2.4.1. Datasets used for optimization and performance evaluation

For optimization, performance evaluation, and performance comparison of HH-MOTiF to other SLiM predictors, all experimentally verified SLiMs from the ELM database as of 26.03.2016 were used (Dinkel et al., 2016). Only SLiMs that have instances in at least three proteins were considered for all the tests, unless specified otherwise. These included a total of 176 ELM classes (also referred to as *groups*) divided into six categories: CLV, DEG, DOC, LIG, MOD, TRG. The classes contain instances of 1,677 unique proteins. However, as some proteins harbor SLiMs from more than one class, the gross number of proteins (i.e., sum of class sizes) is 2,022. These proteins have a total gross length of 1,452,618 residues, out of which 17,909 residues gross belong to the retained ELM classes.

During the optimization of parameters such as minimal and maximal Viterbi scores, minimal and maximal number of columns, minimal length, minimal number of hits, and maximal regex p-value, the two categories CLV and DEG were used as the training set, while the remaining four constituted the test set. The relatively small size of the training set was chosen due to computational restrictions.

2.4.2. Evaluation of low complexity filtering

To shed light on the question, if masking of low complexity regions (LCRs) improves the SLiM search performance, I looked for colocation of LCRs and ELM instances. An LCR was defined as a sequence stretch of the window length length $\boldsymbol{W}$ residues, out of which

at least $\boldsymbol{W}$/2+1 residues are of the same type. For this, I measured the shares of LCR residues in all proteins containing retained ELM classes separately for SLiM- (belonging to at least one of the retained ELM instances) and non-SLiM residues (hereafter referred to as $\boldsymbol{LC_{SLiM}}$ and $\boldsymbol{LC_{nonSLiM}}$, respectively). In addition, the site-wise measure $\boldsymbol{S_{LC50}}$, indicating the share of sites being at least 50% covered by an LCRs, and therefore getting practically non-accessible for the prediction, was computed. Overlapping ELM instances were counted separately.

2.4.3. Evaluation of machine learning-based SLiM-likeliness filtering

2.4.3.1. ELM-based peptide sets

ELM data were used to test the pipelines. For this, the instances of each of the retained ELM classes, were saved as 20-residue long peptides centered on the actual instances; in case of odd instance length, the center was shifted leftwards to the 10th residue. Terminal instances, for which the centered 20-residue long peptides do not exist, were skipped. Three instances in different proteins were selected from each class, in the order they appear in their respective ELM instances file downloaded from the website. To avoid randomization during the selection, the proteins were sorted alphabetically according to their Uniprot identifiers (The UniProt Consortium, 2017). The classes lacking three non-terminal instances were skipped completely to avoid over-representation of certain motifs in the dataset. In total, 468 peptides centered on ELM motifs were generated. These peptides represented the positive set. The positive set was divided into training and test sets. Two different dataset splits were tried out. First, all the ELM classes sorted alphabetically with the odd index (indexing starts from 0) formed the training set, while the classes with the even index formed the test set (the odd-even split). Second, all the classes from CLV, DEG, DOC, and LIG categories formed the training set, while the MOD and TRG classes formed the test set (the 4-vs-2 split). Third, all the classes from CLV and DEG categories formed the training set, while the DOC, LIG, MOD, and TRG classes formed the test set (the 2-vs-4 split). Although instances of different ELM classes can in theory overlap, this was considered as an innate property of the data and the positive test set was not checked for instances already represented in the training set. However, such cases are not frequent for the ELM classes. To form the negative sets, all possible 20-residue long peptides were taken from all proteins containing the retained SLiM instances of the corresponding positive

set. Peptides containing any SLiM residues, including those from non-retained classes, were excluded. After the features were generated from the peptides (see below), duplicating records were removed from the negative sets. In addition, the records already present in the corresponding training set were removed from negative test sets. The exact dataset sizes are provided in Table 5.

Training-test set split	Positive set size		Negative test size	
	Training	Test	Training	Test
Odd-even	234	234	153,799	123,770
4-vs-2	381	87	242,372	33,150
2-vs-4	111	357	73,940	210,716

Table 5. Numbers of peptides in the datasets under two variants of training-test set partition.

2.4.3.2. SLiM feature generation

To form the features, the amino acids were divided in overlapping groups following (Fang et al., 2013) (see Table 6).

Group name	Residue types	Cumulative background probability
Acidic	DE	0.1198
Aliphatic	ILV	0.2019
Aromatic	FHYW	0.0998
Basic	HKR	0.1440
Charged	DEHKR	0.2638
Hydrophobic	ACFILMPVWY	0.4529
P-substrates	STY	0.1591
Polar	DEHKNQRST	0.4783
Small	ACDGNPSTV	0.5037
Tiny	AGS	0.2195

Table 6. The amino acid groups that were used to form the fuzzy dimers potentially important for motif-likeness of a peptide.

The following 434 features were generated:

- Number of occurrences of single amino acids in the peptide (20 features total)
- Number of occurrences of the amino acid groups in the peptide (10 features total)
- Number of occurrences of the dimers consisting of two amino acids belonging to certain groups separated by a linker of the fixed length in the range 0-3 (hereafter referred to as *fuzzy dimers*), e.g., `tiny-x-x-charged` or `polar-hydrophobic` (400 features total)
- Number of stretches of a length at least 2 consisting of the same amino acid type (e.g., `AAACCCCD` counts 2)

- Number of pairs of consecutive positions formed of the same amino acid type (e.g., `AAACCCCD` counts 5)
- Number of different amino acid types represented (e.g., `AAACCCCD` counts 3)
- Maximal number of occurrences of a single amino acid type (e.g., `AAACCCCD` counts 4)

2.4.3.3. Artificial SLiM peptide sets as the control test

A similar procedure to generating the ELM-based peptide sets was chosen. However, instead of selecting peptides out of real protein sequences, 20-residue long peptides were generated, with 250 peptides in each (the training and the test) positive set and 100,000 peptides in each negative set. The negative sets contained exclusively random peptides based on background amino acid probabilities for cytosolic mammalian proteins taken from (Gaur, 2014). The positive set was generated randomly, but with certain patterns in amino acid distribution, which I was then trying to detect. The inserted patterns represented the fuzzy dimers with predetermined linker lengths ***LL*** and thus corresponded to the features that would be generated during the classifier's pre-processing step. In addition, each pattern had its pattern insertion probability ***PIP*** ($0 \leq PIP \leq 1$). To insert a dimer, a random peptide as described for the negative set was first generated. After this, a random number in the range [0,1) was generated and compared with the respective ***PIP***. To proceed with inserting of the current dimer, ***PIP*** had to be greater than the random number. Then, a dimer position was chosen randomly from the available positions. For the first dimer, $19 - LL$ positions are available. For the subsequently inserted dimers, the number of available positions is reduced according to the positions of already inserted dimers. To avoid positional biases at this step, the list of dimers rto insert was reshuffled for each peptide. After choosing the position, the two amino acids are generated. Each amino acid is drawn from the corresponding amino acid group with the relative probabilities inside the group preserved from the background probabilities. For example, if an `acidic` amino acid is generated, only `D` or `E` can be drawn; as the respective background probabilities are 0.0493 and 0.0705, the resulting rescaled probabilities in the group are 0.412 and 0.588. Amino acids selected in this manner replace the original ones from the random peptide.

Even if a certain pattern was not inserted, it still can occur in the random peptide. This pattern occurrence probability is calculated according to the formula:

$$POP = 1 - (1 - P_{left}P_{right})^{19-LL}$$

$\boldsymbol{P_{left}}$ and $\boldsymbol{P_{right}}$ denote the cumulative amino acid group background probabilities for the dimer and are taken from Table 6.

2.4.3.4. The classification pipeline

The classification pipeline was constructed as a complete machine-learning model, which was then executed on all the datasets described. The model was independently, without information leakage from one set to another, fit to the data of the training sets and then applied on the respective test sets. As a consequence, relative feature importance was determined for each training-test set pair individually.

To build the machine learning model, the features were first pre-processed, while the feature values were scaled so that each feature has the mean value 0 and the standard deviation value 1. The features with a constant value, if there were any, were pruned away.

Then, the univariate feature selection procedure was carried out to leave $\boldsymbol{k}$ best features out of those that survived pruning. Support vector machines (SVM) with the radial basis function (RBF) kernel with gamma 0.01 and slack $\boldsymbol{C}$ were used to carry out binary classification of the peptides on the basis of retained features.

The hyperparameters $\boldsymbol{k}$ and C were being optimized in a nested cross-validation (CV) procedure with subsequent validation on the corresponding test set, which was effectively an out-of-sample test set. In the outer CV layer, the positive test set was selected as a random continuous block of the original positive training set so that it contained approx. 20% of the ELM classes. In this manner, the new training and test set did not contain SLiM instances from the same classes. The respective negative sets were formed through random choosing without replacement of the corresponding number of elements from the original negative training set, so that the numbers of negative and

positive elements are equal for each run of the classifier. Finally, the inner CV layer was a simple random split with 20% of the outer CV training set forming the inner CV test set. For the outer CV, 50 splits were performed, while the inner CV had 10 splits. The high number of splits for the outer CV was justified, as each split involved not only partitioning of the set but also undersampling of the negative set. The inner CV layer was needed, as a single training round proved to be extremely prone to over-fitting, which was reflected in a high discrepancy between the training and test scores. With the nested CV, the inner and outer test scores were much closer to each other, although the outer level still showed somewhat lower mean and significantly larger variance (see Figure 6).

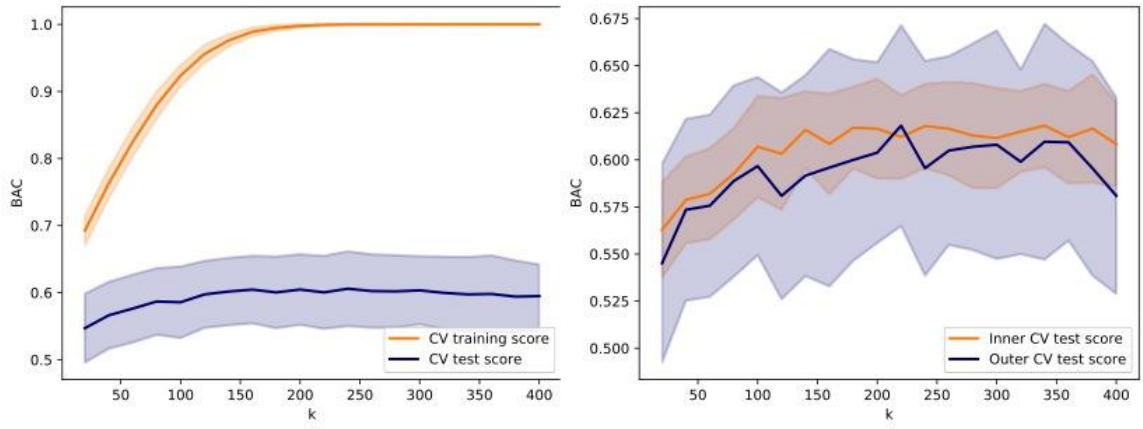


Figure 6. Balanced accuracy (BAC) of the SVM classifier (RBF, C=1, γ=0.01) in the single-layer (left) and the nested (right) cross-validation as dependency of the number k of best features retained. 20% test set; outer layer: 50 splits, inner layer: 10 splits. The dataset used is the training set of the odd-even split of the ELM dataset (see main text). The lines show the average value, the bands show the standard deviation.

Optimization of the SVM hyperparameters C and γ was straightforward, as the outer CV scores showed the clear optimal value γ=0.01 (see Figure 7) and the strong boundary between C<1 and C≥1 (see Figure 8). Nine values in the logarithmic space were tried for both C and γ. More charts like those shown were examined to draw the conclusion on the optimal values. The trend in C was consistent across different values of γ, and vice versa.

Optimization of k turned out to be more complicated, as there was no clear optimal value across different values of C and γ. For example, the combination C=1, γ=0.1 suggested that the classifier can operate only with small k, while the combination C=1, γ=0.001 demonstrated positive correlation of performance and k. The selected optimal

combination suggested rather an optimum around ***k***=200; however, this tendency was not stable upon repeating the procedure several more times. Therefore, I made the decision to continue optimizing ***k*** for each dataset individually.

The resulting classification pipeline retains an element of randomness, as individual training runs were performed on randomly chosen subsets of the input dataset, which includes undersampling the negative set.

The resulting machine learning models, which included the list of retained features with their scaling transformations, as well as the SVM hyperplane; all random ones up to a certain degree were tested on the corresponding out-of-sample test sets. No undersampling of the negative sets was performed during this test. Due to the randomness, the procedure was repeated 20 times and the mean and standard deviation were calculated.

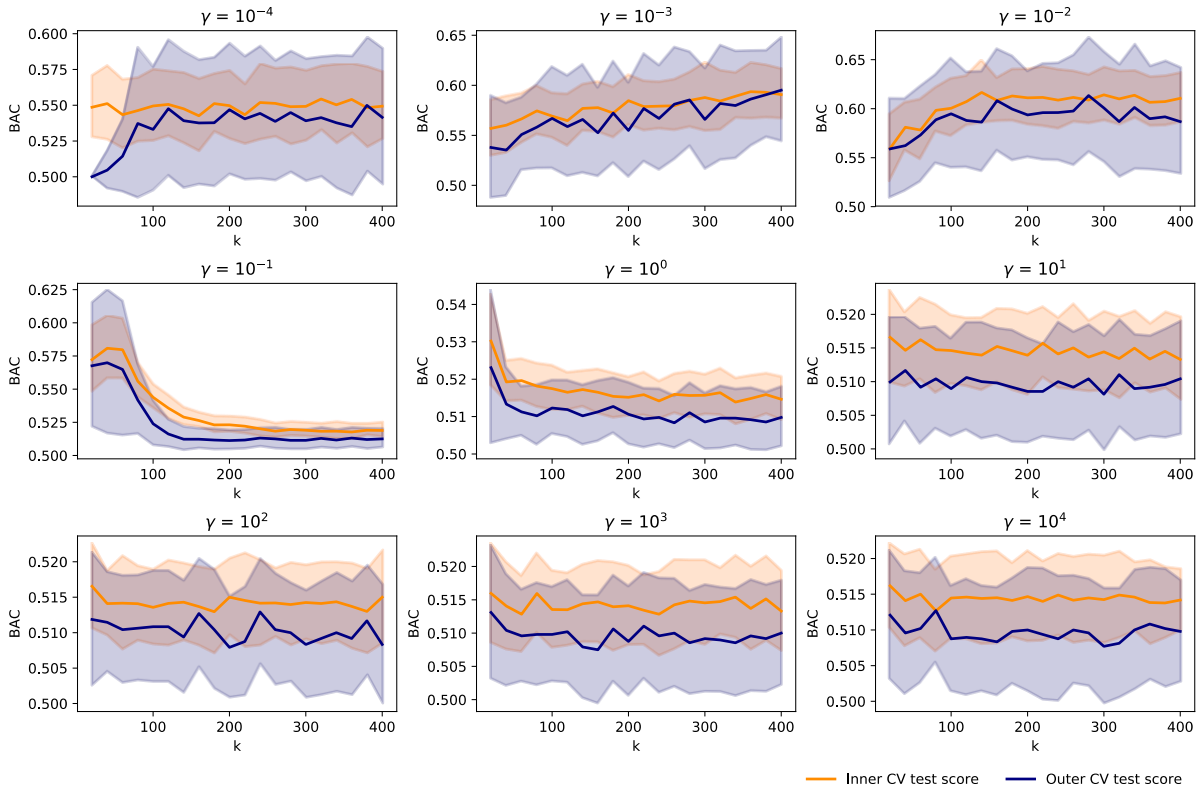


Figure 7. Balanced accuracy (BAC) of the SVM classifier (RBF, ***C***=1) in the nested cross-validation as dependency of ***γ*** and the number ***k*** of best features retained. The dataset used is the training set of the odd-even split of the ELM dataset (see main text). The lines show the average value, the bands show the standard deviation.

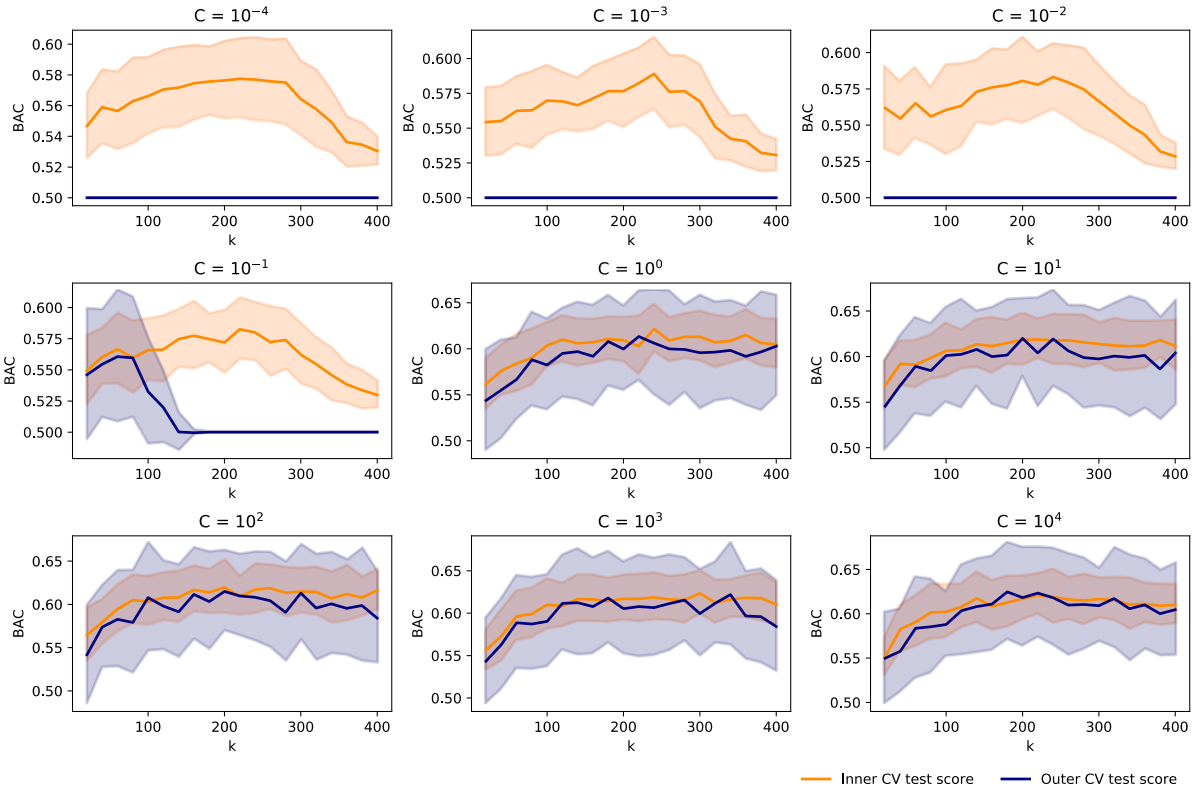


Figure 8. Balanced accuracy (BAC) of the SVM classifier (RBF, γ=0.01) in the nested cross-validation as dependency of C and the number k of best features retained. The dataset used is the training set of the odd-even split of the ELM dataset (see main text). The lines show the average value, the bands show the standard deviation.

2.4.4. Estimation of HH-MOTiF result quality with an automated procedure

At the early stages of development, an estimator for the quality of the prediction results was written. This estimator fulfilled the following requirements: 1) it provided a confidence score in the absence of an external reference, as is the case when really unknown SLiMs are predicted. 2) it chose the best results from multiple alternatives.

All motif leaves were classified as reciprocal or non-reciprocal. Reciprocity is marked in the pseudo-MSAs of the current version of HH-MOTiF, although the classification does not influence the tree scoring or filtering in any way. A reciprocal leaf overlaps with the root of another motif tree by at least three residues. In this case, a 'reciprocity score' R is calculated as the share of the overlapping residues of another tree among all motif leaves. This score was used as a confidence score for the results.

Before the statistical model for deriving the minimal number N_{min} of query proteins to contain a SLiM from the total number of queries N and the homology filter were

introduced, choosing the right N_{min} for a given dataset represented a challenging task. Therefore, different values, which depended on shares/percentages of N, were tried. For relatively large datasets (N>50), the N_{min} values of the share in the range [0.05, 1.00] with the step 0.05 were tested. The actual N_{min} was calculated as products of N and the share; N_{min} has been rounded upwards. In addition, different thresholds for the Viterbi score were tried, whereby scores from 6 to 17 with a step size of 0.5 were tested. As a result, a parameter matrix was being generated. The estimator had to pick the best combination of all tested parameters with the highest R.

2.4.5. Calculation of performance metrics (the SLALOM algorithm)

The performance estimator is implemented as an independent Python program named SLALOM. The earlier versions were tuned specifically for the evaluation and comparison of HH-MOTiF, SLiMFinder (Edwards et al., 2007), MEME (Bailey et al., 2006), and GLAM2 (Frith et al., 2008). The recent version, however, is a generalized version for working with arbitrary kinds of positional annotations of a grouped collection of sequences. It was initially adapted from the work of Song and Gu (Song and Gu, 2015), but then became more similar to Bioconductor (Lawrence et al., 2013) and bedtools (Quinlan, 2014). The current version is, however, expanded beyond the possibilities of these in evaluating performance of SLiM predictors.

To make the description of the algorithm more general, I refer to an instance of a SLiM – predicted or annotated – as *site* (the term was coined by Song and Gu in (Song and Gu, 2015)). Speaking generally, a site is a continuous sequence element characterized by its start and end positions in a specific sequence. Moreover, for further generalization, I introduce the term *symbol* in the meaning of a minimal unit the sequence consists of. While speaking about protein sequences, I continue to refer to symbols as residues. In further examples of application of SLALOM – concerning DNA and time series – symbols become base pairs and time units, respectively.

SLALOM takes two different annotations of sites in the same collection of sequences. In the case of a SLiM predictor evaluation, these annotations are the ELM database (the standard) and the output of the predictor.

In the SLALOM algorithm, the following basic measures are calculated residue-wise: TP: true positives: residues being present in both, the golden standard and the prediction; FP: false positives: residues predicted but not annotated in the standard; FN: false negatives: annotated in the standard but not predicted; and TN: true negatives: present neither in the standard nor in the prediction. However, all numbers except for TN will be different depending on the way of resolving overlaps: the counts may be residue-resolved, which means that a residue is counted only once, even if it is a part of multiple sites; or gross, if a residue is counted as many times as it occurs. The schematic example is shown on Figure 9. As a result, FP and FN both split to residue-resolved and gross, while TP splits to three different measures: residue-resolved TP, which are residues that are present at least once in both the standard and the prediction, $\boldsymbol{TP_{rr}}$; and two gross TPs: residues gross in the standard that are also present in the prediction, $\boldsymbol{TP_{gross,a;}}$ and residues gross in the prediction that are also present in standard annotation, $\boldsymbol{TP_{gross,p}}$. These measures are calculated as total residue counts in all the input sequences. Residue-resolved counts sum up to the total length of the sequences; gross measures, in sum or separately, can exceed the total length. If there are no overlaps, residue-resolved and gross measures are equal.

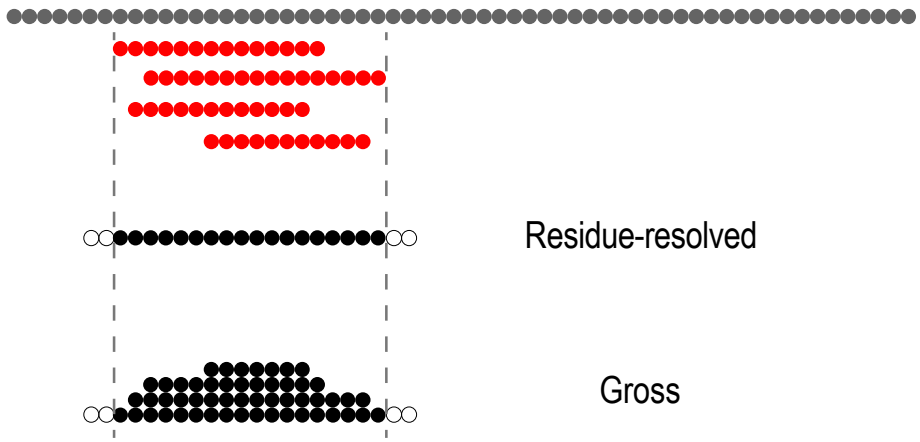


Figure 9. Schematic example illustrating the difference between residue-resolved (symbol-resolved) and gross counting. The gray circles mark residues in an input sequence. The red circles depict overlapping annotated sequence elements within this sequence. The black circles illustrate the residues counted by SLALOM. The hollow circles indicate that nothing was counted at this position. The figure was submitted as part of a publication to *BMC Bioinformatics*.

Based on these measures, the following statistics are calculated:

- Recall (a.k.a. true positive rate, TPR, or sensitivity):

$$Rc = \frac{TP}{TP + FN}$$

- Precision (a.k.a. positive predictive value, PPV):

$$Pr = \frac{TP}{TP + FP}$$

- Specificity (SPC):

$$Sp = \frac{TN}{TN + FP}$$

- False positive rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

- Performance coefficient (PC):

$$PC = \frac{TP}{TP + FP + FN}$$

- Accuracy (ACC):

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

All of these measures can be calculated as residue-resolved and as gross. It makes more sense to calculate the gross recall with $\boldsymbol{TP_{gross,a}}$ and the gross precision with $\boldsymbol{TP_{gross,p}}$. The gross PC and ACC are even more ambiguous. Therefore, SLALOM does not calculate gross ACC. For calculation of gross PC, $\boldsymbol{TP_{gross,p}}$ is used. Note that PC with this definition deviates slightly from the original calculation for HH-MOTiF as described in (Prytuliak et al., 2017).

The same measures are also calculated site-wise. In this case, however, the counts are always gross and, as TN is undefined, specificity and false positive rate cannot be calculated. Moreover, in order to classify predicted sites into TP and FP, as well as benchmark sites into TP and FN, one needs to set the criteria, when one considers a predicted and an annotated site as a match. These matching criteria define, to what

extent a benchmark and a predicted site have to overlap to consider them as a match. In SLALOM, a match is only considered, when two sites overlap by at least ***R*** residues and/or ***P***% of the site length. It is also possible to select, from which of the two sites these ***P***% should be taken, if their length is different: the *current* (the benchmark site for the recall calculation, the predicted site for the precision, etc.), the *shortest* or the *longest*. In addition, one can try simultaneous matching of several closely located, predicted sites to a single benchmark site, and vice versa. This is referred to as a *patched* overlap logic: one site can be *patched* with several small sites, so that they meet the ***R*** and ***P*** criteria cumulatively. A schematic example is shown on Figure 10.

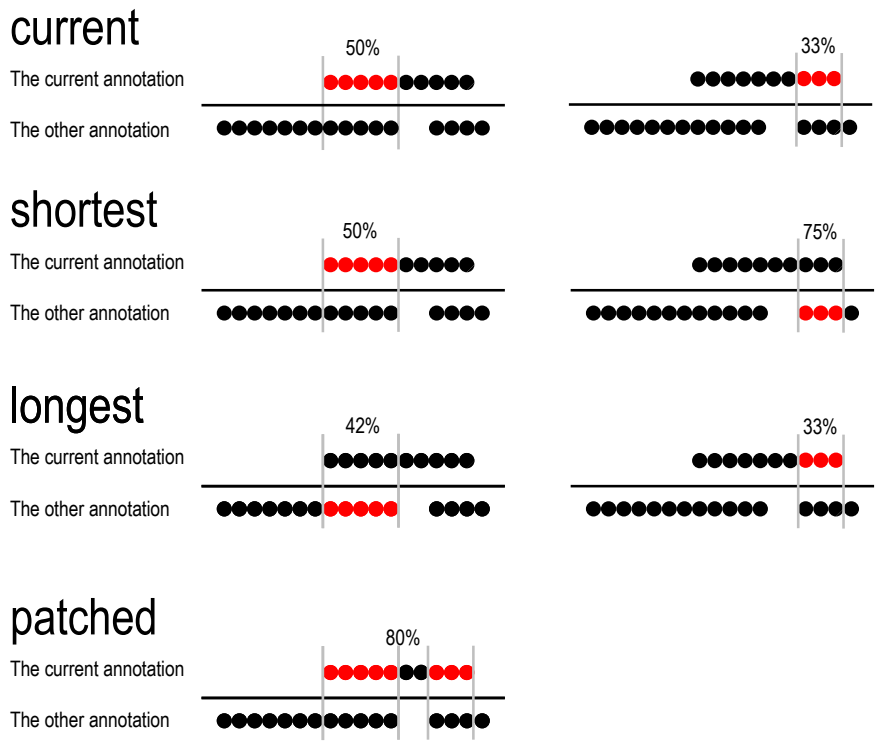


Figure 10. Schematic example illustrating differences between four site matching logics supported in SLALOM. The input annotation pair is parsed twice, so that each of the annotations becomes the current one once. Each site in the current annotation is evaluated separately to be classified as either having a match in the other annotation or having no match. In this example, the current site is 10 residues/symbols long and it overlaps with two sites in the other annotation: one 12-residue-long and the other 4-residue-long. If the *current*, *shortest* or *longest* logic is chosen, the current site is evaluated against each site in the other annotation it overlaps with separately (i.e., one site at a time) with subsequent choice of the best candidate, while under the *patched* logic all the sites in the other annotation are implicitly combined and a cumulative match is assessed. The figure was submitted as part of the SLALOM publication to *BMC Bioinformatics*.

On the basis of these measures, additional measures are calculated. The following are relevant for the examples provided in this thesis:

- Informedness (In):

$$In = Rc + Sp - 1$$

- F1 score:

$$F1 = \frac{2 * Rc * Pr}{Rc + Pr}$$

Note that in the publication (Prytuliak et al., 2017) the balanced accuracy was used instead of informedness. These metrics are related to each other through a linear transformation. The reason for the replacement is that random performance will result in the balanced accuracy of 0.5 and the informedness of 0.0. The latter is more consistent with the corresponding value of the F1 score, which was chosen as the main metric for performance evaluation, optimization, and comparison.

SLALOM is designed to handle *grouped* collections of sequences. For example, in the particular case of the HH-MOTIF evaluation, a group is all the sequences containing SLiMs from an ELM class. The groups may be overlapping. All the metrics are calculated for each group separately. The counts TP, FP, FN, and TN are simply summed up for sequences in each group before any metrics are calculated. After this, the metrics are simply-averaged across all the groups to produce the final metric for the whole ELM dataset. This approach is hereafter referred to as *group-wise averaging*. Alternatively, the counts (TP, etc.) can be calculated for each sequence separately, divided by the sequence length, and only then summed up within the group with the rest of the procedure being unchanged. This approach is hereafter referred to as *sequence-wise averaging*. During the evaluation of HH-MOTiF, the group-wise averaging was applied.

It should be noted that sometimes a zero division is encountered during the calculation of certain metrics. This happens when one of the annotations does not contain any sites in the given sequence group. For example, when a SLiM predictor does not return any candidates for the given ELM class, the TP and FP counts are both zeros. In this case, the zero division is encountered while calculating the precision. If the zero division is encountered, the metric value is by default set to `nan` and it is not considered for subsequent averaging across the dataset. Alternatively, the `nan`s can be treated as zeros,

generally reducing the average value. The former logic was applied while measuring the precision of HH-MOTiF and other SLiM predictors. However, for measuring the natural F1 score (see below), `nan`s were treated as zeros during the averaging.

Finally, the overall F1 score can be calculated as average of the F1 scores for each group as well as by calculating F1 based on averaged recall and precision. The former is hereafter referred to as the *natural* F1, while the latter is referred to as the *synthetic* F1. The natural F1 is the value returned by the SLALOM program and it was used as the internal optimization goal throughout the development of HH-MOTiF. The synthetic F1, however, was reported in the publication (Prytuliak et al., 2017) for the reasons of consistency with the approach of Song and Gu (Song and Gu, 2015), who reported the synthetic F1.

2.4.6. Statistical model for false positive predictions and tests on negative data

A false positive prediction in the benchmarking context is a putative SLiM returned by a predictor but absent in the benchmark database. Such a prediction occurs, if there are similar enough sequence stretches in the input protein set. These stretches may represent homologous regions, a yet unknown SLiM or just a random coincidence of amino acid composition. I assume that HH-MOTiF can successfully filter out the homologous regions and the probability of getting an unknown real SLiM in a small set of randomly selected proteins is negligibly small. Random occurrences, however, are a serious problem, given the shortness and weak conservation of real motifs.

A rigorous *ab initio* model would require an assessment of the whole sequence space of a motif that theoretically can be detected by HH-MOTiF, as it was done for the SLiMBuild algorithm in (Edwards et al., 2007). However, unlike SLiMFinder, HH-MOTiF does not explicitly construct a motif space internally. Instead, it constructs the motif trees from pairwise alignments outputted by `hhalign` in a multi-step procedure. Assessing the motif space would require the knowledge on the distribution of the alignments and their associated scores for real protein sequence pairs, as well as careful evaluation of the impact of each of the steps of the procedure.

Therefore, a simpler alternative of the *ab initio* model was developed. I call it a posterior model, as it estimates the probability of an observed false positive prediction. As outlined in (Gelman et al., 1996), there are three general ways to evaluate such a model. The first way is to check for the robustness of the model, i.e. how sensitive it is to small changes in the initial assumptions. The second way is the rigorous analytical check of the model logic under the given context. The third way is a goodness-of-fit test to check, how the model fits the real data. In this work, the third way is chosen.

The goal of developing this model is to calculate a minimal number $\boldsymbol{N_{min}}$ of proteins required to share a SLiM on the basis of the dataset size $\boldsymbol{N}$. Too low $\boldsymbol{N_{min}}$ will lead to excessive false positives, while too high one will discard too many true positives.

At the beginning I made a preliminary estimation $\boldsymbol{N_{prel}}$ of a reasonable $\boldsymbol{N_{min}}$:

$$N_{prel} = \max(ceil(0.3 * N), 3)$$

Then, I measured the actual residue-wise FPR (residue-resolved) for random, negative data sets of different sizes. A negative set of size $\boldsymbol{N}$ is constructed from shuffled proteins containing ELM SLiMs, whereby only one protein per ELM class can belong to the same set. 100 negative sets of six different sizes (600 sets in total) were constructed. All the predictions from these data sets were considered false positives, even if they are occasionally overlapping with ELM instances. The first round of measurements was done using $\boldsymbol{N_{prel}}$ as $\boldsymbol{N_{min}}$.

After this, I modeled the FPR with the following posterior model:

Let us assume that the shared ambiguous sequence stretch that led to the formation of a given false positive SLiM candidate has the probability $\boldsymbol{P_1}$ to occur by chance in a typical protein sequence. *Ambiguous* means that this stretch might be described with ambiguous, optional and/or wildcard positions; however, it still must contain enough information to pass the score threshold. Then, the probability to find this stretch in $\boldsymbol{N_{min}}$ independent sequences is

$$P_n = P_1^{N_{min}}$$

Let us furthermore assume that we observe most of the trees in exactly N_{min} proteins, although they potentially can locate to N_{min}+1,…,N proteins. This assumption is reasonable as long as P_1<<1; it is also confirmed by the manual inspection of randomly picked false positives trees. As N_{min}<=N, the observed false positive motif tree could have potentially located also to some other subset of N_{min} proteins within a given N-set. To continue, one has to estimate the expected number E of trees that could have been formed in the given N-set from a stretch with probability P_1 to occur in a single protein. Here, it would be tempting to calculate E as $C_{N_{min}}^{N} * P_1^{N_{min}}$. However, this would only be correct for the case of prior probabilities, while a posterior model is being developed. In the considered case, it is already known that the stretch is present in the specific N_{min} proteins. Therefore, the resulting expected number of trees is in fact the probability that some of the observed instances could have been 'flipped' to other N-N_{min} proteins of the N-set. The probability to find the stretch in one of these other proteins is P_1, while the corresponding probability for the N_{min}-subset is 1. Thus, the formula is:

$$E = \sum_{k=0}^{\min(N_{min}, N-N_{min})} C_k^{N-N_{min}} * P_1^k$$

If no 'flipping' is possible (when N_{min}=N), the E is expectedly 1. Moreover, E grows with growing N for a fixed N_{min}, while E decreases with growing N_{min} for a fixed N. This is the intuitively expected behavior.

Finally, the FPR can be modeled by applying the Šidák multiple testing correction formula (Wright, 1992):

$$FPR = 1 - (1 - P_n)^E$$

The resulting formula is a function of N, N_{min}, and P_1. The latter reflects some generalized protein property not dependent on the particular set. Thus, P_1 is also dependent on neither N nor N_{min} and, once estimated from a particular case, be applied to all other cases. Let us estimate P_1 for the case N_{min}=N=3. The formula in this case simplifies to:

$$FPR_{3,3} = P_1^3$$

From the negative set testing, it was observed that $\boldsymbol{FPR_{3,3}}$=0.017. This leads to $\boldsymbol{P_1}$=0.26. This value then can be applied to estimate FPR for other values of $\boldsymbol{N}$ and $\boldsymbol{N_{min}}$=$\boldsymbol{N_{prel}}$. These values are shown in Table 7. Note that these values were measured at a preliminary version of HH-MOTiF, before introducing the $\boldsymbol{N_{min}}$ estimation as well as certain other modifications, so that they must not exactly correspond to the corresponding FPR values of the current version. The current version demonstrates lower FPR values.

$\boldsymbol{N}$	$\boldsymbol{N_{prel}}$	Observed FPR	Modeled FPR
3	3	0.017	0.018
5	3	0.023	0.028
10	3	0.095	0.083
15	5	0.012	0.012
20	6	0.010	0.008
25	8	0.003	0.001

Table 7. Observed and modeled FPR for HH-MOTiF with $\boldsymbol{N_{prel}}$ as $\boldsymbol{N_{min}}$ applied to negative sets (100 randomly generated sets of each size).

From Table 7, it can be concluded that the model accuracy is acceptable for the purposes of being used in HH-MOTiF. Therefore, it was added to HH-MOTiF to evaluate the FPR for different alternative $\boldsymbol{N_{min}}$ for a given $\boldsymbol{N}$. The lowest $\boldsymbol{N_{min}}$, for which the predicted FPR is below 1%, is chosen as the final $\boldsymbol{N_{min}}$. The exception is $\boldsymbol{N}$=4: in this case, the FPR is slightly above 1% for $\boldsymbol{N_{min}}$=3; $\boldsymbol{N_{min}}$=4 turns out to be too strict, suppressing virtually all true positive predictions as well. Therefore, the former gets chosen, overriding the model decision.

$\boldsymbol{N}$	$\boldsymbol{N_{min}}$	Observed FPR	Modeled FPR
3	3	0.018	0.015
5	4	0.006	0.004
10	5	0.004	0.003
15	6	0.002	0.001
20	6	0.008	0.009
25	7	0.005	0.009

Table 8. Observed and modeled FPR for HH-MOTiF with $\boldsymbol{N_{min}}$ proposed by the specially developed statistical model applied to negative sets (100 randomly generated sets of each size).

To prove that the model works as expected, the FPR measurements for the proposed $\boldsymbol{N_{min}}$ values were conducted. Again, 600 random negative sets were generated. The procedure was repeated also for $\boldsymbol{N}$=3 and $\boldsymbol{N}$=20, although there was no change in $\boldsymbol{N_{min}}$

for these. As can be seen from Table 8, all values, except the one for ***N***=3 (and ***N***=4), lie below the threshold of 1%. After the model was finished, other modifications were introduced to HH-MOTiF. These modifications further reduced the observed FPR values.

2.4.7. In-depth performance analysis

To spot possible opportunities for improvement of HH-MOTiF, as well as to provide more comprehensive comparison with other available tools, an in-depth performance analysis was conducted. This analysis consisted of several independent parts.

First, the weighted averages for the selected performance metrics were calculated. By default, all the groups (i.e., ELM classes) have the weights of 1 during the database-wide averaging. I additionally tried weighting by the total number of proteins, sites, and residues (the total length of the proteins) in the class. The weights themselves are available in Supplementary Table S10 of (Prytuliak et al., 2017). The number of sites is equal to the number of proteins for the majority of the classes; however, the former exceeds the latter if there are multiple SLiM instances in the same protein.

Second, the classes (groups) were classified by the number of proteins they contain to detect, if there is a dependency of the tools' performance on the dataset size. The classes with less than three proteins had been already filtered out and therefore were not considered. The following categories were formed according to the class sizes: 3-5, 6-10, 11-15, 16-25, 26-50, 51+, and 3-50 member proteins. The synthetic residue-wise F1 was calculated for each cluster as the performance measure.

Third, different site matching criteria were applied to calculate the recall and precision of the HH-MOTiF prediction of ELM motifs. The minimal numbers of residues ***R***=1;3;5 were tried. For ***R***=1, the minimal length percentages ***P***=0;25;50;75 were tried. For each combination of ***R*** and ***P***, all four possible ways to choose the site to apply the ***R*** and ***P*** were tried: *current*, *shortest*, *longest*, *patched* (explained in Section 2.4.6. Calculation of performance metrics (the SLALOM algorithm)).

Fourth, different overlap resolving and averaging logics were tried to study their impact on the performance of HH-MOTiF. Three distinct logic-defining parameters were tried:

gross vs. residue-resolved counting, group-wise vs. sequence-wise averaging, and setting the zero division results to `nan` vs. zero. As a result, eight different logic combinations were tested. Residue wise recall, precision, and FPR were computed for each of them.

2.4.8. Comparison with existing tools

The performance of HH-MOTiF was compared with that of standalone versions of SLiMFinder (Edwards et al., 2007) (v. 5.2.3), MEME (Bailey et al., 2006) (v. 4.0), and GLAM2 (Frith et al., 2008) (v. 4.11.1). In addition, data on the performance of whmm (Song and Gu, 2015) were provided for comparison. In this case, published data were used. All the tools were compared on the whole ELM dataset (176 classes) using the same performance metric calculation procedure. The following metrics were calculated: residue-wise recall (residue-resolved), precision (gross), synthetic F1, natural F1 (gross), PC (gross), specificity (residue-resolved), and informedness (residue-resolved); site-wise recall, precision, synthetic F1, natural F1, and PC. The data on whmm lack natural F1, specificity, and informedness; furthermore, it is not clear from the publication, if residue-wise recall and PC are calculated as gross or residue-resolved.

The command lines to run SLiMFinder, GLAM2 and MEME respectively with the optimized configurations:

```
slimsuite/tools/slimfinder.py seqin=input_file.fasta resdir=output_directory/ resfile=output_directory/output.csv dbtype=prot walltime=24.0 blast+path=/usr/bin iupath=/home/roman/apps/iupred/iupred dismask=T consmask=T probcut=1.0 topranks=5
```

```
glam2 -a 3 -b 15 -O output_directory/ p input_file.fasta
```

```
meme input_file.fasta -oc output_directory/ -minw 3 -maxw 15 -nmotifs 5 -protein -maxsize 1000000
```

To be able to use the same standalone performance evaluation tool with all the predictors, the output of the predictors was pre-parsed with dedicated Python scripts to produce CSV files of the same format.

3. RESULTS

3.1. Application scenarios of HH-MOTiF

To predict SLiMs in proteins *de novo* is a pressing problem in bioinformatics – and biology. Many research groups aim to map interaction sites in proteins, which can subsequently be mutated, for instance to create specific loss of function mutants. HH-MOTiF was developed to match this demand in the scientific community, which is not well covered by existing software solutions. In HH-MOTiF, I automated the procedures that were performed manually during earlier collaboration projects for SLiM detection. The goal was to develop a tool that would fill the gap in basic SLiM search scenarios. There are four such scenarios, on which I focused my development efforts:

1. Finding a potential, yet unknown SLiM in a set of 4-10 proteins of diverse length and composition that are experimentally supported candidates to have a common biological function (e.g., binding to the same interaction partner or shared subcellular localization).
2. Finding a conserved SLiM, which, however, moves along the sequence during the evolution. The input in this case is a collection of distant orthologs or paralogs of the same protein.
3. Finding enriched SLiM candidates in a larger number of proteins (500-2000) identified, for example, in the course of a proteomic study.
4. Finding already known SLiMs, which are not yet annotated in publicly available databases; these can for instance be SLiMs that were identified in the course of a motif search in one of the first three scenarios.

Scenario 1 has become the main application of HH-MOTiF. Scenario 2 is also covered by the functionality of HH-MOTiF; however, we lack a collection of experimentally verified datasets to reliably evaluate the performance of HH-MOTiF under these conditions. Scenario 3 can be handled by the HH-MOTiF back-end application, which allows command-line application of the tool: owing to the long processing times, the size of the input dataset in the web-server is limited to 50 proteins. Scenario 4 is implemented as the proteome-wide search option of the HH-MOTiF web-server.

3.2. The HH-MOTiF web-server

HH-MOTiF is publicly available as web-server under the address http://hh-motif.biochem.mpg.de. Neither registration nor providing personal data is required to access the full functionality of the server. Cookies and additional plugins are also not required.

The web-server consists of four pages: the main *Search* page with data submission forms; the *About* page with the short description of the algorithm and the application scope; the *Guide* page with explanation on input formatting, usage, and output interpretation; the *Tests* page with additional examples of datasets that can be processed with HH-MOTiF. In addition, there is a page with the *Contact* information of the authors.

The main page contains two submission forms: one for the *de novo* search and one form for the proteome-wide search (see Figure 5). *De novo* search can be done in two modes: **standard** and **advanced**. The **standard** mode requires only the input dataset and will start a SLiM search with the optimized parameters, as they were published in (Prytuliak et al., 2017).

The **advanced** mode allows the user to modify some parameters – switching on/off sequence masking (the surface accessibility, homology, and sequence disorder filters), as well as adjusting the maximal regex p-value – as well as to submit his/her own collections of orthologs instead of relying on the automatic reciprocal BLAST procedure. The **standard** mode accepts the dataset either as a FASTA file or as FASTA-formatted input in the text field. The **advanced** mode accepts either a FASTA file or a ZIP archive of FASTA files – one file for each query protein with its orthologs; if the FASTA file is submitted or the ZIP archive consists only of FASTA files with a single sequence, and the automatic orthology search is switched off, then evolutionary sequence conservation will not be assessed. In advanced mode, the user also has the possibility to submit a masking file, which will be merged with other selected masking options. The gap restriction option will discard SLiM candidates with gaps longer than one residue in their alignments. The option "*Show best suboptimal if no motifs found*" will display the best identified motif tree, even if it does not satisfy the minimal Viterbi score or the regex p-value criteria, or if it does not pass the alignment recognition filter.

For searching proteome-wide, the user has to submit the sought-for motif as a FASTA-formatted alignment and choose the organism to look in. The option "*Only full-length matches*" will suppress hits that do not have full-length alignments with the input motif profile.

All three modes have the optional field for the e-mail address to notify the user, when the job is complete. Furthermore, the user can provide an optional name to the job, which may be useful to distinguish between the results from different submissions.

After pressing the "***Submit***" button, the user's input data are getting checked for validity. If the check fails, the user is redirected to the submitted form with the corresponding error message displayed. From the error log written so far, I observed that the most typical user errors are caused by trying to submit too few (usually only one) or too many (more than 50) proteins, as well as by submitting aligned FASTA files instead of the required non-aligned ones.

The *Guide* page is divided into five sections describing in detail the input and output of both, the *de novo* and the proteome-wide search in addition to general information.

The *Tests* page highlights five input datasets (ELM classes) that in my opinion most clearly demonstrate the advantages of HH-MOTiF over other available tools. These advantages are the ability to carry out motif searches also in homologous query proteins, as well as to distinguish low complexity SLiMs from unrelated LCRs. The corresponding exemplary output web pages look exactly like the real HH-MOTiF outputs, but with additional highlighting of the real ELM instances. This makes it easier to estimate the performance of HH-MOTiF and double check the author's calculations.

3.3. SLALOM – a statistical method for positional data comparisons

As the performance evaluation, optimization, and comparison of HH-MOTiF with other tools turned out to be an ambiguous tasks with no solution described in the available literature, I developed SLALOM – a statistical method to carry out these tasks. SLALOM

emphasizes some sources of ambiguity that may arise during a comparison of a pair of positional annotations.

Here, I outline the SLALOM's scope of applications followed by the description of the identified sources of ambiguity.

SLALOM can compare a wide range of types of positional annotations represented as lists of sites in a grouped collection of sequences. The sequences may be of any kind; however, the two annotations must relate to the same sequence. When comparing a pair of such annotations, there are two mutually exclusive, cumulatively exhaustive situations possible:

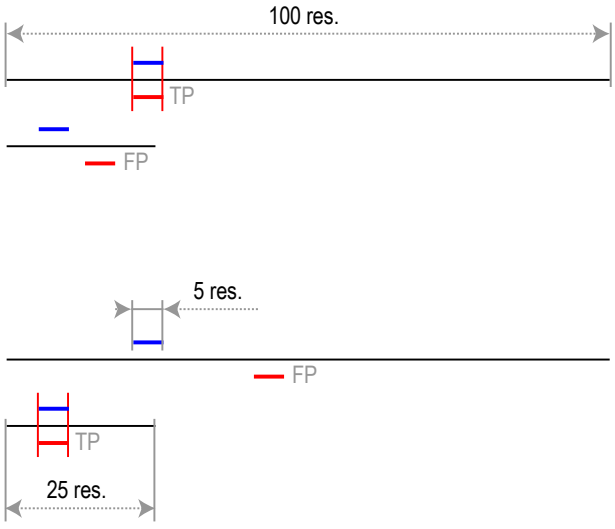
1. The annotations concern the same type of sites, e.g., when comparing one annotation of motif against another annotation of motifs. In this case, the expected ideal situation would be complete identity of them.
2. The annotations concern different types of sites, e.g., when associating genes with their promoters in DNA sequences or different types of events in time series. In this case, the two annotations are not expected to be identical. In fact, they often will not overlap at all; however, when a respective null hypothesis is true, they must relate to each other according to some rules.

SLALOM can deal with both situations. For this, different statistical metrics are calculated. Residue-wise accuracy (ACC), informedness, F1 score, or Matthew's correlation coefficient (MCC) will be in most cases suitable for the situation 1. Site-wise recall and precision in combination with total site counts will be normally the most informative metrics in the situation 2. It is important to note that SLALOM can also match non-overlapping sequence elements (e.g., genes and promoters) just by proximity. The produced results in the TSV format can be assessed manually or piped to another script or a statistical software package for in-depth statistical analysis and/or plot building. Furthermore, in scenario 1, the two annotations can either have the same level of reliability, when comparing the output of two predictors with each other; or one can be more reliable than the other, when comparing the output of a predictor with a verified, 'golden standard' annotation. The latter is also referred to as benchmarking.

These two cases are distinguished in SLALOM through providing different sets of performance metrics in the output.

SLALOM was designed to specifically deal with four sources of ambiguity:

1. Resolving overlapping sites within one annotation. If a residue/symbol is traversed by more than one site, it can be counted in several different ways.
2. Matching sites between the two input annotations. It must be specified which degree of overlap is sufficient to register a match. Moreover, the degree of overlap itself can be computed in different way, especially if the sites have different lengths.
3. Diversity in site length. As the similarity or performance can be measured residue-wise, longer sites may outweigh the shorter ones. This may be desired or not, depending on the research question.
4. Diversity in sequence length. If the site length and number do not depend on the sequence length, the positive and negative residues (symbols) become more unbalanced in longer sequences. Because of this, some metrics become highly dependent on the order of averaging. A schematic example is shown in Figure 11.



	Sequence-wise		Group-wise	
	TP	FP	TP	FP
	5%	0%	5 res.	0 res.
	0%	20%	0 res.	5 res.
average	2.5	10	2.5	2.5
precision	20%		50%	

	0%	5%	0 res.	5 res.
	20%	0%	5 res.	0 res.
average	10	2.5	2.5	2.5
precision	80%		50%	

Figure 11. Schematic example of the impact of the averaging order on performance metrics. Sequence-wise averaging is sensitive to distribution of true positives (TP) and false positives (FP) among the input sequences, while group-wise averaging is not. In this example, two alternative predictions (red lines) of the same real sequence elements (blue lines) in a dataset consisting of two sequences (100 and 25 residues long) ate illustrated. All the sequence elements – both real and predicted – are 5 residues long. If sequence-wise averaging is applied, the prediction, which correctly identifies the element in the shorter sequence (lower panel) is scored better than the alternative prediction (upper panel): the corresponding

precision values are 80% and 20%. On the other hand, with group-wise averaging both predictions score 50%. The figure was submitted as part of the SLALOM publication to *BMC Bioinformatics*.

To ensure the correct treatment of each of the sources of ambiguity applicable to the given dataset, SLALOM offers the user a number of options to control the input data preprocessing and subsequent calculations (for details, see Section 2.4.6. Calculation of performance metrics (the SLALOM algorithm)).

SLALOM is publicly available for download as a standalone software package at https://github.com/BCF-calanques/SLALOM.

3.4. Performance of HH-MOTiF in comparison with other tools

To estimate the added value of HH-MOTiF to the field of *in silico* SLiM prediction, I compared HH-MOTiF with other available tools in the task of recovering ELM motifs, whereby a total 176 motifs was analyzed. The tools that are most cited in the recent literature and which are available for download as a standalone version were used for comparison. These include MEME (Bailey et al., 2006), GLAM2 (Frith et al., 2008), and SLiMFinder (Edwards et al., 2007). The results are summarized in Table 10. In (Prytuliak et al., 2017), we also included the tool whmm, as it also employs HMM comparisons (Song and Gu, 2015). Unfortunately, the downloadable version of the tool did not work in our hands. We therefore calculated performance values in a way that ensures compatibility with the results published in (Song and Gu, 2015) and (Song et al., 2015): we have chosen the synthetic F1 as the main performance metric. Some tested SLiM predictors perform quite differently with different parameter sets. SLiMFinder, for instance, shows variable performance with and without the statistical model evaluating the motif space. Therefore, I also calculated performance values for other configurations of the selected tools (see Table 11).

Tool		Residue-wise							Site-wise				
		TPR	PPV	SPC	F1n	F1s	PC	In	TPR	PPV	F1n	F1s	PC
HH-MOTiF	training set	0.2072	0.4321	0.9932	0.2014	0.2801	0.1725	0.2004	0.2212	0.5319	0.2315	0.3125	0.2176
	test set	0.2107	0.4185	0.9924	0.1963	0.2803	0.1583	0.2032	0.2386	0.5683	0.2499	0.3361	0.2213
	overall	0.2102	0.4202	0.9925	0.1970	0.2802	0.1602	0.2028	0.2363	0.5638	0.2475	0.3330	0.2209
SLiMFinder	default	0.1324	0.6337	0.9989	0.1398	0.2190	0.1317	0.1313	0.1546	0.6698	0.1570	0.2512	0.1469
	optimized	0.2034	0.3500	0.9917	0.1940	0.2573	0.1715	0.1951	0.2715	0.3893	0.2357	0.3199	0.2080
GLAM2	default	0.4902	0.0433	0.7109	0.0734	0.0796	0.0429	0.2011	0.5126	0.1925	0.2614	0.2799	0.1857
	optimized	0.3731	0.0839	0.8750	0.1220	0.1370	0.0822	0.2481	0.4162	0.1839	0.2301	0.2551	0.1782
MEME	default	0.0440	0.0257	0.9850	0.0298	0.0324	0.0216	0.0107	0.0481	0.0779	0.0531	0.0595	0.0469
	optimized	0.2192	0.0609	0.9363	0.0881	0.0953	0.0547	0.1555	0.2490	0.0989	0.1324	0.1416	0.0878

Table 10. Comparison of performance metrics for different tools and configurations tested. TPR: true positive rate, sensitivity, recall; PPV: positive predictive value, precision; SPC: specificity; F1n: natural F1

score, average of F1 scores for individual ELM classes; F1s: synthetic F1 score, harmonic mean of the averaged Rc and Pr; PC: performance coefficient: In: informedness. The table is based on the data published in (Prytuliak et al., 2017).

The two tables show that the performance was significantly different not only between different tools but also between different configurations of the same tool. Moreover, the performance varied to a different extent for different metrics. Some configurations could recover more ELM SLiMs – resulting in higher recall – , while others had a higher share of positive occurrences in their predictions – showing higher precision. Precision also expectedly correlated with specificity.

Tool		Residue-wise							Site-wise				
		TPR	PPV	SPC	F1n	F1s	PC	In	TPR	PPV	F1n	F1s	PC
HH-MOTiF	I	0.214	0.375	0.992	0.197	0.272	0.159	0.206	0.242	0.508	0.250	0.328	0.222
	II	0.240	0.370	0.990	0.214	0.291	0.175	0.230	0.269	0.490	0.266	0.347	0.237
	III	0.300	0.265	0.969	0.213	0.281	0.164	0.269	0.342	0.374	0.286	0.357	0.238
	IV	0.233	0.258	0.989	0.221	0.245	0.175	0.222	0.268	0.351	0.288	0.304	0.251
SLiMFinder	V	0.101	0.615	0.999	0.113	0.174	0.104	0.100	0.118	0.661	0.128	0.200	0.120
	VI	0.142	0.680	0.999	0.153	0.235	0.142	0.141	0.166	0.722	0.174	0.270	0.163

Table 11. Performance metrics for additional configurations of HH-MOTiF and SLiMFinder that differ from the default configurations in the following ways: I – with disorder masking in addition; II – with surface accessibility masking replaced by disorder masking, III – with homology filtering disabled, IV – with the option 'show best suboptimal if no motifs found' activated, V – with surface accessibility masking (NetSurfP RSA with cutoff 0.16), VI – with disorder and conservation masking (providing the options '`dismask=T consmask=T`'). TPR: true positive rate, sensitivity, recall; PPV: positive predictive value, precision; SPC: specificity; F1n: natural F1 score, average of F1 scores for individual ELM classes; F1s: synthetic F1 score, harmonic mean of the averaged Rc and Pr; PC: performance coefficient: In: informedness. The table is based on the data published in (Prytuliak et al., 2017).

Neither a high recall nor a high precision by itself is indicative of a good SLiM predictor. A recall of 100% can be reached by marking whole query sequences as SLiMs. The perfect precision can be reached by exactly predicting only a few instances of one – the most conserved and easiest to identify – SLiM, while completely ignoring all the others. The latter will also achieve highest specificity, although this can be achieved even in an easier way, namely by returning no SLiMs at all. However, neither of these predictors will be useful, despite scoring 100% in certain metrics. Furthermore, as SLiMs mark only a small share of proteins sequences (on average, 1.2% in the ELM dataset), we deal with strongly unbalanced positive and negative elements. This means that also high (approx. 98.8%) accuracy (ACC) can be easily achieved with an algorithm returning always no SLiMs. Therefore, I have formulated two requirements for candidates for the main statistical metric: 1) It should achieve its maximal value if and only if a perfect prediction is given; and 2) its value should be close to zero for random or useless predictions.

I have chosen the F1 score, which is the harmonic mean of recall and precision and satisfies the requirements as the main statistical metric. To clarify ambiguities in the order of averaging, I further defined the synthetic and natural F1 scores. In addition, I calculated informedness and performance coefficient (PC) that also satisfy the requirements.

Although I used the F1 score as the main metric, I have provided and considered different metrics to understand the advantages and disadvantages of each tool and configuration. Based on my observations, I have come to several conclusions: 1) GLAM2 systematically overpredicts SLiMs, which helps to achieve high recall, but comes at the cost of low precision. As the dataset is imbalanced, this allows maintaining specificity at a reasonable level (>80%), which also translates in high informedness; 2) SLiMFinder can work in two different modes: in the first mode, the number of SLiMs in the output is limited by the significance threshold. In this mode, high precision is achieved, although at the cost of low recall. SLiMFinder returns no results for 127 out of 176 ELM classes with settings optimized to achieve high F1. Precision can be increased even further (up to 80% residue-wise) through the activating of more rigorous statistical modeling (the option '`-cloudfix=T`'); however, the recall drops even more, causing also the F1 score to fall substantially. In the second mode, the number of outputted SLiM candidates is always fixed. I found five to be the optimal value. As SLiMFinder often outputs overlapping SLiMs, the returned set of five SLiMs may however still effectively represent only one or two SLiM candidates. Therefore, with some ELM classes, SLiMFinder demonstrates high precision also in this mode. Nevertheless, the averaged precision is significantly lower in this case. Generally, this second mode of SLiMFinder resembles the performance profile of HH-MOTiF. HH-MOTiF was from the beginning designed to achieve performance balanced in terms of recall and precision. As a consequence, HH-MOTiF is not able to match neither GLAM2 in recall nor SLiMFinder in precision, but performs better than the other tools, if a harmonic evaluation is performed.

3.5. In-depth analysis of the performance of selected SLiM predictors

Besides providing the averaged F1 score for evaluating the overall performance of a tool, one can obtain much more information on fine details of the performance of a predictor

by conducting additional procedures. First, one can Compare values of different metrics between each other. Second, one can compare values of the same metrics for different subgroups of the input dataset. Finally, one can compare values of the same metrics computed with different logics (overlap resolving, order of averaging, etc.).

In the previous section, some conclusions through comparing different metrics, namely the recall and precision, were discussed. I have also looked at the differences between residue-wise and site-wise performance. Site-wise metrics judge the ability of a predictor to roughly hit the positions of annotated SLiMs in the database – i.e. the benchmark dataset. Residue-wise metrics evaluate in addition the ability to correctly predict the lengths and exact positions of SLiMs. Therefore, residue-wise metrics are generally expected to be lower than the corresponding site-wise ones, although, if the SLiM instances (real or predicted) are highly diverse in length, the opposite may also become true.

One can indeed see that the site-wise performance was always better than the residue-wise performance. The magnitude of the differences, however, varies between the tools. For example, both precision values of SLiMFinder are quite similar. This suggests that when this predictor finds a true SLiM, it is also able to avoid predicting too many flanking residues of the SLiM. The differences are somewhat larger in recall. Taken together, SLiMFinder tends to under-predict SLiMs, and the returned SLiMs are shorter than annotated. The situation is opposite for GLAM2: its site-wise recall only marginally exceeds the residue-wise one, while the differences in precision are much larger. This indicates that GLAM2 tends to rather over-predict SLiMs, capturing excessive flanking residues along with the core SLiMs. Both MEME (with optimized settings) and HH-MOTiF have slightly elevated site-wise recalls and approx. 1.5-fold greater site-wise precisions indicating a slight over-prediction of SLiMs.

Next, I have looked at results for subgroups of ELM motifs. The division in subgroups of the ELM dataset can be performed in many ways. For example, Song and Gu divide the ELM dataset into six parts according to its inherent categories: CLV, DEG, DOC, LIG, MOD, TRG (Song and Gu, 2015)(Song et al., 2015). I have also looked at the performance of HH-MOTiF across these categories (see Table 12) and observed that performance on the

SEG and LIG categories was substantially better than on the CLV and DOC categories. I also saw that the performance differences between the categories reflected consistently on both recall and precision, both residue- and site-wise. The exception was the MOD category, which had an exceedingly high precision, both residue- and site-wise. The ratios between corresponding site-wise and residue-wise metrics also remained relatively constant, further showing that the abovementioned over-prediction of SLiMs by HH-MOTiF was consistent across the motif categories. The good performance of HH-MOTiF on the DEG category may be also explained by its belonging to the training set: it was used for training, and subsequent fine-tuning of the algorithm. I also measured the category-wise performance for other tools (see Table 13). Intriguingly, SLiMs from DEG, LIG, and TRG categories were easier to identify for all the tools, including whmm – see Figure 3 in (Song and Gu, 2015). On the contrary, the CLV, DOC, and MOD categories proved to be difficult to handle. This observation led me to the preliminary conclusion that some motifs are objectively harder to predict than others, independent of the method used. Nevertheless, there are exceptions to this trend. For example, the performance of SLiMFinder on DOC motifs exceeded its performance on LIG ones. This suggests that some algorithms are better suited than others to identify specific SLiMs. These issues are more specifically discussed in a later part of this thesis (Section 4.4. SLiM heterogeneity as the challenge to statistical evaluation of predictors).

In addition to the functional classification of SLiMs provided by ELM, I have also divided the ELM classes into subgroups according to various criteria. One such criterion was the dataset size, which represents the number of unique proteins a specific SLiM is encountered in. The data are summarized in Table 14. Only SLiMFinder showed a clear dependence on the dataset size.

Motif				Residue-wise						Site-wise		
category	TPR	PPV	SPC	F1n	F1s	PC	In	TPR	PPV	F1n	F1s	PC
CLV	0.103	0.250	0.997	0.092	0.146	0.074	0.100	0.103	0.371	0.112	0.161	0.104
DEG	0.244	0.473	0.992	0.240	0.322	0.207	0.236	0.263	0.568	0.274	0.360	0.258
DOC	0.080	0.211	0.996	0.065	0.116	0.053	0.076	0.083	0.290	0.079	0.129	0.073
LIG	0.265	0.424	0.990	0.243	0.326	0.196	0.255	0.302	0.580	0.314	0.397	0.278
MOD	0.119	0.519	0.997	0.135	0.194	0.105	0.116	0.130	0.650	0.161	0.217	0.133
TRG	0.163	0.399	0.996	0.151	0.231	0.129	0.159	0.188	0.581	0.190	0.284	0.178

Table 12. Performance of HH-MOTiF across six ELM categories. TPR: true positive rate, sensitivity, recall; PPV: positive predictive value, precision; SPC: specificity; F1n: natural F1 score, average of F1 scores for individual ELM classes; F1s: synthetic F1 score, harmonic mean of the averaged Rc and Pr; PC: performance coefficient: In: informedness.

Motif category	Dataset count	Residue-wise synthetic F1			
		HH-MOTiF	SLiMFinder	GLAM2	MEME
CLV	6	0.146	0.005	0.026	0.000
DEG	17	0.322	0.428	0.215	0.175
DOC	19	0.116	0.258	0.059	0.020
LIG	96	0.326	0.236	0.149	0.112
MOD	21	0.194	0.155	0.079	0.081
TRG	17	0.231	0.390	0.182	0.199

Table 13. Performance of different tools and optimized configurations across different ELM categories.

Dataset size	Dataset count	Residue-wise synthetic F1			
		HH-MOTiF	SLiMFinder	MEME	GLAM2
3-5	61	0.236	0.114	0.074	0.148
6-10	48	0.297	0.267	0.097	0.129
11-15	24	0.232	0.352	0.069	0.084
16-25	29	0.310	0.314	0.104	0.144
26-50	12	0.407	0.442	0.225	0.169
51+	2	0.434	0.059	0.056	0.137
3-50	174	0.279	0.259	0.095	0.126
All	176	0.280	0.257	0.095	0.137

Table 14. Performance of different tools (the optimized settings) subgroups of the whole ELM dataset according to the size of individual sub-datasets. The table is based on the data published in (Prytuliak et al., 2017).

3.6. Initial application of automated results parsing and optimization

This section describes an algorithm, which was initially implemented in HH-MOTiF but was later phased out due to discovery of better alternatives. Nevertheless, the work on this algorithm has enabled other improvements in HH-MOTiF. For this reason, these initial attempts are briefly discussed here.

The output produced by an earlier version of HH-MOTiF was very bulky. As alignment recognition and some other filters were not yet implemented, a much higher number of motif trees were present in the output. For the same reason, the average number of motif leaves per tree was higher and the leaves themselves were longer. As a result, the output file size for larger datasets could reach several gigabytes. In addition, the core HH-MOTiF algorithm has one parameter, for which a valid parameter value was yet undefined: the number of query proteins that need to contain the SLiM. Different values of this parameter were tried to select the intuitively best output. The generalized optimal value could not be defined, as it was varying heavily for different input datasets. As it was challenging to process such amounts of data manually, I implemented an *in situ* optimization procedure to parse different alternative outputs and select the intuitively best one automatically. The procedure was launched each time a dataset was processed.

The initial goal of the *in situ* optimization was to maximize the internal quality criteria – the reciprocity score ***R***. The data for the three test datasets formed out of all proteins harboring SLiMs in distinct ELM categories are presented in Table 15.

The idea of maximizing ***R*** was based on the assumption that for strong SLiM candidates, multiple instance can assume the role of a motif root, making the corresponding motif leaves reciprocal. The idealization of this situation is illustrated by the ELM class TRG_LysEnd_APsAcLL_3, which is the sample motif used on the HH-MOTiF web-server. There are three motif trees found in this 3-protein dataset; each motif root overlaps with the leaf of both other trees and each motif leaf overlaps with the root of another tree. As all the overlaps are no less than three residues, the reciprocity score is 1 for this dataset. On the other hand, if the leaves are mostly not reciprocal, there is a high chance that the motif tree is formed randomly. Further tests proved this assumption to be correct only for relatively large datasets and low values of ***R***. The latter is based on the observation that upon substantial over-prediction of SLiMs, high reciprocity is achieved by chance. Ultimately, if every single residue in the dataset is predicted to be in a motif root, the reciprocity is always 1. The measurement of correlations between residue-wise F1 score and ***R*** for different ranges of ***R*** showed that the optimal value for ***R*** is close to 0.2 (see Table 16). Therefore, I redefined the optimization goal as minimization of |***R***-0.2|.

Dataset	Residue-wise F1 score	
	Maximal	Automatically chosen
CLV	0.050	0.025
DEG	0.256	0.090
DOC	0.053	0.041

Table 15. Performance of an earlier versions of HH-MOTiF on the category-wide datasets with the automated *in situ* optimization. The Maximal F1 score refers to the parameter configuration that would be optimal for the specific dataset.

Range of ***R***	Dataset				
	CLV	DEG	DOC	MOD	TRG
[0, 1]	0.46	-0.36	-0.33	0.15	-0.11
[0, 0.2]	-0.40	-0.83	-0.55	-0.46	-0.77
[0.2, 1]	0.74	0.66	0.38	0.54	0.38

Table 16. Pearson correlation coefficients between residue-wise F1 and the reciprocity score for different parameter configurations of HH-MOTiF on the category-wide datasets.

Yet, the correlation between ***R*** and the performance was observed only for relatively large (50+ proteins) and noisy datasets, in which the sought-for SLiMs are present only in part of the queries. As the typical use case of HH-MOTiF was rather small datasets, the *in situ* optimization was ultimately phased out. Such an approach had obvious speed disadvantages caused by a need to launch the SLiM search several hundred times with different parameters with subsequent result analysis. However, intermediate data caching allowed running the whole search in only approximately twice the time of a single independent search. On the other hand, auto-choosing the parameters each time is less prone to the danger of over-fitting in comparison to approaches, where optimal values for the parameters are derived in a training procedure as a compromise between optimal values for individual subsets in the training set.

However, the idea of adaptive results was implemented in the option of showing the best suboptimal hit, if no results are found with default parameters: activating this option forces turning off some of the filters and recalculating the results if no SLiM candidates are found initially. As expected, activating this option increases recall and decreases precision of HH-MOTiF. However, the increase in recall is quite small (residue-wise 0.233 vs. 0.210 for the default configuration) and the net effect on synthetic F1 is markedly negative (0.245 vs. 0.280). This means that the implemented filters are tuned very well to the task; turning them off results in retrieving many more additional false positives but rescuing only few false negatives. Nevertheless, this option is left for the user's consideration, as seeing the best suboptimal hit may give a better idea of what can be found in the submitted dataset with the proposed method.

3.7. Other biological applications of SLALOM

The statistical method, SLALOM, is not limited to a certain type of positional annotation data. Therefore, besides evaluating and comparing the output of SLiM predictors, it can be applied to a wide range of problems, whenever comparison of positional annotations of continuous sequence elements (also referred to as sites) in a set of sequences is required.

SLALOM is applicable to comparing annotations of various sequence elements (sites) as well as sequence properties, which are assessed residue-wise. Besides SLiMs, sites

include conserved domains in proteins or genes in genomes. Residue-wise sequence properties include the degree of disorder or surface exposure in proteins and propensity for point mutations in genomes. SLALOM can compare annotations of the same elements or properties originating from different sources. It assesses their similarity or reliability. In addition, its ability to match heterogeneous elements can answer, among others, question about the percentage of motifs located in transmenbrane regions or the extend to which disordered or exposed regions overlap.

SLALOM has been used to compare genome annotations from two different sources, namely GenBank (Benson et al., 2013) and RefSeq (Pruitt et al., 2012). The genome of the archeon *Natronomonas pharaonis* was used for this comparison. Its GenBank annotation contains 2,694 protein-coding genes, while its RefSeq annotation contains only 2,608 genes. Each gene is assigned to one of the six reading frames, three frames in each strand. Besides the fact that some genes are missing in one of the annotations, those that do exist in both sometimes do not match perfectly. As the two databases use different identifiers, it is not possible to unambiguously determine, if the identical gene is annotated, should two genes only overlap partially. Initially, a threshold of 50% of the *current* gene length was used. The key comparison statistics are provided in Table 17.

If gene matching is tried regardless of the reading frames, differences in symbol-resolved and gross counts were observed, indicating the presence of overlaps. However, these were gone upon discrimination of the reading frame: the 4 still overlapping base pairs in the RefSeq genome could be attributed to an overlap with a pseudo-gene. This means that almost all the overlaps occur only in different reading frames, which is an expected situation.

Measure	Without frame separation		6 distinct reading frames	
	Symbol-resolved	Gross	Symbol-resolved	Gross
GenBank total gene length	2,359,033	2,367,915	2,367,915	2,367,915
RefSeq total gene length	2,336,204	2,339,349	2,339,345	2,339,349
Genbank genes matched	2,589		2,583	
Refseq genes matched	2,596		2,587	
Symbol-wise ACC	0.9861	-	0.9967	-
Symbol-wise F1	0.9923	0.9923	0.9892	0.9892
Site-wise F1	0.9779		0.9751	

Table 17. Statistics on comparing two annotations of the *Natronomonas pharaonis* genome. A pair of sites/genes is matched if >=50% overlap of the current site is reached.

Furthermore, there was a rise in ACC upon frame separation. This rise can be explained by the false positive paradox, which influences some metrics on unbalanced datasets. A genome is an example of an unbalanced dataset. The genes in all frames cover ~90% of the whole DNA length (2,595,221 base pairs). However, for each frame, the coverage is only ~15%. As the number of mismatches in this specific case is roughly proportional to the gene length, a higher ACC is observed when averaging ACC from all the frames separately, as compared to measuring ACC for all the frames together. The F1 score, on the other hand, is a more robust metric and is not subject to the false positive paradox. The F1 score dropped upon frame separation, because the matches between genes in different frames are gone.

Moreover, we could see that different numbers of genes were matched in different genomes, which means that some matches are not unequivocal. This is not a desired situation, when mapping features from one genome to another. Therefore, stricter matching criteria were tested for registering a match between a pair of genes in different annotations. As shown in Table 18, a higher threshold of overlap of the current gene does not solve the problem of non-reciprocal matches: when one annotation contains a shorter gene that is completely covered by a longer one in the other annotation, the number of non-reciprocal matches can only increase with the minimal overlap criteria, as the shorter gene is always covered to 100%. However, when applying the minimal overlap percentage to the longer gene, one gets the full reciprocity already with 50%; increasing the criterion will further eliminate dubious matches until only perfectly matched genes are left. For further analysis, SLALOM offers the option to output only unmatched ('`-os_diff unmatched`') or only non-perfectly matched ('`-os_diff discrepant`') sites.

Measure	Minimal overlap to match a pair of genes				
	Of the current gene		Of the longest gene		
	50%	90%	50%	90%	100%
Genbank genes matched	2,583	2,555	2,580	2,528	2,400
Refseq genes matched	2,587	2,562	2,580	2,528	2,400
Site-wise F1	0.9751	0.9651	0.9732	0.9536	0,9053

Table 18. Statistics on comparing two annotations of the *Natronomonas pharaonis* genome. The reading frames are treated separately.

3.8. Analysis of time series data with SLALOM

SLALOM was developed as a general-purpose software for any kind of positional annotations in a collection of sequences. There are no specific requirements concerning the type of elements (symbols) the sequences consist of. These must not necessarily be alphabetical, such as proteins and DNA. A sequence can also be a time series, where each symbol represents a time unit, e.g., a minute or a day. The annotated elements in this case represent events with defined start and end time points. The time stamps provided by the user are converted to sequence positions, representing distances to the sequence start, so that the time series are internally treated in the same way as the protein sequences. Note: The very first element has the position 1.

Time series analysis is useful in medical studies, where one needs to correlate onset times of symptoms or psychological conditions (see for instance (Nielsen et al., 2017)) in a group of patients or to determine possible reasons for increased mortality during an epidemic (Nunes et al., 2011).

Here, the application of SLALOM on economic time series is shown. We looked at possible correlations between economic news releases and the movements in the EURUSD exchange rate.

First, one needs to define an exchange rate movement. I looked into two separate categories of movements. All the movements were assessed on the basis of the prices at the end (a.k.a. closing price) of consecutive time intervals with fixed duration (one such interval is also called a candlestick). The first movement category is referred to as a *trend:* it is defined as at least five consecutive 5-minute candlesticks, where each interval, including the first one, is closing at the consistently higher or lower price than the previous one. The second category is *a spike*: it determines a series of consecutive 1-minute candlesticks that satisfies the two following criteria:

1. The difference between the closing prices of the last candlestick in the series and the last candlesticks before the series is at least 0.002 USD
2. Each candlestick has consistently higher or lower closing price than the previous one by at least 0.0005 USD.

The trends and spikes were marked as time intervals (by start and end time points) with a specially written Python script on the basis of data downloaded from HistData.com. In the calendar year 2015, there were 1,801 trends and 426 spikes with the average durations 29 and 2 minutes, respectively.

The news releases were marked correspondingly as time intervals beginning at the release time point and ending after 30 or 2 minutes, respectively to approximately match the duration of a trend or a spike. The data on the news releases were downloaded from FXStreet.com. SLALOM can work with the downloaded files directly with no need of additional pre-processing. In this example, I grouped the news releases by country. Data from seven countries were considered, out of which three form the control group, i.e. have neither EUR nor USD as domestic currency. Sometimes, there were several releases announced simultaneously in a certain country. These I counted as a single release. While analyzing trends, non-simultaneous releases, although not frequent, could also tightly follow each other, so that the corresponding 30-minutes intervals overlapped. In this case, I considered only the last release in the series, as the effects of the preceding releases, if there were any, may have been distorted: if there were positive news followed by negative ones for EUR, the beginning upward trend could be interrupted and reversed into a downward one; or if there were non-significant news closely followed by important ones, the impact of the latter could be mistakenly associated with the former.

In this example, I considered news releases, as well as movements as atomic events and therefore focused on the site-wise statistics. Note that the duration (length) was not important. Upon analysis of releases, they were classified in two categories, depending on whether the following 30- or 2-minute interval coincided with a movement. A similar classification was carried out for movements. To avoid non-specific matches with trends, I registered the coincidence if and only if it spanned at least 50% of the duration of the release or trend; as the spikes lasted on average only 2 minutes, any coincidence was registered as a match in the analysis of spikes.

The operating mode of SLALOM should be chosen on the basis of the data interpretation. In this example, the movements were treated as a given fact, while the news releases

were considered as a putative predictor of upcoming movements. Therefore, SLALOM was launched in the benchmarked mode, with movements being the first annotation (the benchmark). As a result, the site-wise TPR was the share of matched movements, while the site-wise PPV was the share of matched news releases. Consistent with the assumed causality, I also tried limiting matches to cases, in which movements actually start not earlier than the news release: this means that the predictor has the 'leading' nature. However, as a release-driven enhancement of an already developing subtle movement in the same direction will be overseen with the current trend definition, this additional criterion may create false negatives. The results – with and without the criterion – are presented in Table 19.

Country	Number of news releases	News releases as trend predictor				Ratio 'leading' / 'any order'	
		Any begin order		Release is leading			
		TPR	PPV	TPR	PPV	TPR	PPV
United States	857	0.0866	0.1628	0.0489	0.0910	0.565	0.559
China	139	0.0133	0.1633	0.0083	0.1020	0.624	0.625
Japan	387	0.0316	0.1587	0.0217	0.1108	0.687	0.698
Germany	240	0.0272	0.2125	0.0150	0.1250	0.551	0.588
United Kingdom	331	0.0361	0.1971	0.0200	0.1088	0.554	0.552
France	154	0.0144	0.1753	0.0056	0.0714	0.389	0.407
Italy	179	0.0155	0.1639	0.0105	0.1093	0.677	0.667

Table 19. Performance of economic events as a predictor of trends in EURUSD throughout the calendar year 2015.

Without performing exact statistical tests, one can notice that the data obtained imply the independence of news releases and price movements. Approximately seven 29-minute trends per trading day mean that the market in 2015 was trending ~15% of the time, which explains the share of matched news releases well: the market was trending equally often regardless of a news release. Furthermore, the share of matched trends was linear with the number of releases, which strengthens the last argument. There was also no visible difference between the impact of news releases in the control and EUR/USD countries. Indeed, the news releases from UK seemed to influence the EURUSD rate more than the events in France or USA. The only exception represented Germany, where 51 out of 240 releases coincided with trends. Though, whether this number is statistically significant is hard to tell. With the background probability 0.173,

which is the average for the control countries, the binomial test yielded the p-value 0.065. After a multiple testing correction for 4 countries:

$$p \approx 1 - (1 - 0.065)^4 \approx 0.24$$

This means that also for Germany, the results hardly can be viewed as significant.

The ratios between the numbers for the 'leading' releases and the corresponding numbers for both leading and non-leading ('any order') releases additionally strengthen the assumption of independence. The duration of both releases and trends is around six 5-minute intervals on average. Therefore, if at least 50% overlap is required, there are seven possible relative positions expected: a trend starts 15 minutes earlier, 10 minutes earlier, and so forth until 15 minutes later than the release (see Figure 12). Four out of them are leading releases (i.e., satisfy the requirement of the trend starting not earlier). Thus, the expected ratio under the independency assumption is 4/7 (≈0.57), which is consistent with the observed ratios.

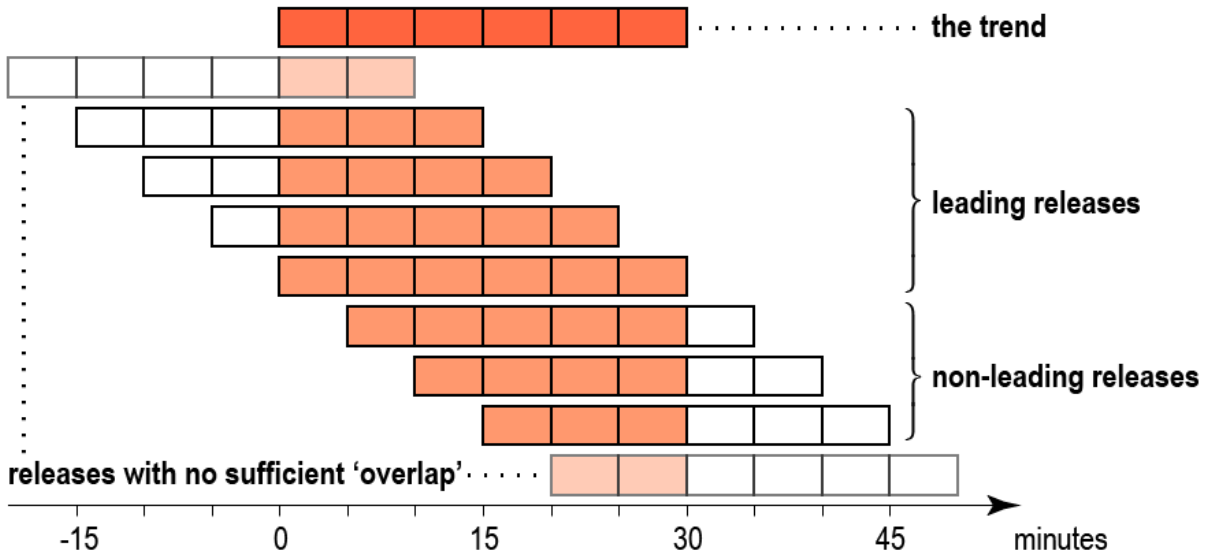


Figure 12. Schematic representation of relationships between theoretically possible news releases and a 30-minute trend. Each rectangle depicts a 5-minute interval. At least 50% 'overlap' is required to register a match between a news release and the trend. A release is leading if and only if it 'begins' no later than the trend.

The data for spikes are presented in the Table 20. The metrics were measured with the 'leading' requirement, as in the case of spikes, the minimal price difference per candlestick was imposed, which makes it much less probable for a random movement to

become a part of a spike. As one can see from the TPR numbers, there were only two spikes out of 426 that coincided with the news releases in USA, only one for Germany and Italy, and no single one for France. I consider this fact to be strong evidence for the assumption of independence between spikes and news releases.

Country	Number of events	Site-wise TPR	Site-wise PPV
United States	1185	0.0094	0.0025
China	152	0.0000	0.0000
Japan	415	0.0047	0.0048
Germany	241	0.0047	0.0083
United Kingdom	341	0.0469	0.0528
France	158	0.0000	0.0000
Italy	195	0.0047	0.0103

Table 20. Performance of economic events as a predictor of spikes in EURUSD throughout the calendar year 2015

3.9. Application of machine learning for evaluating SLiM-likeness

In this section, I discuss the performance of the SVM-based pipeline in discriminating between SLiM-containing peptides and peptides containing no SLiMs annotated in ELM. Hyperparameters of the pipeline were optimized only for the odd-even dataset partition; the same parameters were used for all other runs. The results are summarized in Table 21. There is an expected tendency of rising performance with the rising size of the training set.

Training-test set split	BAC		***k***	
	Mean	SD	Mean	SD
Odd-even	0.595	0.012	180	78
4-vs-2	0.594	0.028	213	103
2-vs-4	0.540	0.017	171	104

Table 21. Performance on out-of-sample test sets for different splits of the ELM-based peptide dataset. BAC: balanced accuracy. The mean and SD are based on ***n***=20 tests.

To assess the amount of information the SLiM-likeness is based on, as well as the potential maximal performance of this kind of approach, I also tested the pipeline on artificial, randomly generated peptides. This test also served as the control to check, if the demonstrated above-random performance is not a computational artifact. The results are summarized in Table 22. As one can see from the table, strong patterns, which exactly correspond to one of the analyzed features (series A), ensured the high classification accuracy with low variance. 100% accuracy could not be achieved, as some peptides from the negative set contain the pattern just by chance. The performance

expectedly dropped, when some of the peptides from the positive set did not contain the pattern (series B against A and series E against D). Weaker patterns, which almost certainly occur just by chance, showed drastically weaker performance (series D against A and series D against J). Nevertheless, when the same weak pattern was repeated several times, the performance rose again (series H), which was also an expected behavior, as the current classification pipeline is based on feature occurrence counts and not on the binary occurrence detection. Some of the inserted patterns did not correspond exactly to any analyzed feature. Namely, the series C presents the linker length 4 that is not covered in the feature space. The series F, on the other hand, represents the case of not predefined amino acid groups, `[ST]` and Proline. Nevertheless, these are still subgroups of, for example, P-substrates and Hydrophobic respectively. This is no longer the case for the groups in the series I and K: Every Fifth, First Five, Navy, and Vowels – see Table 21 – are neither predefined amino acid groups nor subgroups of those. However, the classification was still possible, although with a lower accuracy (series C against A and series I against J). When there are different patterns inserted simultaneously, the performance expectedly grew (series L against I and K). Finally, the accuracy was expectedly close to 50%, when completely random positive datasets were used (series G) or the pattern to be found was too weak (series E).

Group name	Residue types	Cumulative background probability
Every Fifth	FLRY	0.2227
First Five	ACDEF	0.2472
Navy	ANVY	0.1922
Vowels	AEIY	0.2097

Table 21. Artificial amino acid groups that were used to test the ability of the proposed machine learning pipeline to detect differences in features not explicitly checked for. All these groups are not part of the groups formed on the basis of biophysical and biochemical properties and presented in Table 6.

Series	Inserted patterns	***PIP***	***POP***	BAC		***k***	
				Mean	SD	Mean	SD
A	acidic-aromatic	1.0	0.2043	0.896	0.005	32	17
B	acidic-aromatic	0.5	0.2043	0.681	0.008	25	9
C	acidic-x-x-x-x-aromatic	1.0	0.1651	0.664	0.009	237	96
D	polar-small	1.0	0.9947	0.514	0.013	182	129
E	polar-small	0.5	0.9947	0.501	0.011	263	105
F	[ST]P	1.0	0.1524	0.716	0.018	44	42
G	Completely random positive set			0.498	0.008	246	107
H	polar-small	1.0	0.9947	0.711	0.016	78	64
	polar-small	1.0	0.9947				

			0.9947				
	polar-small	1.0					
I	vowels-x-x-first_five	1.0	0.5954	0.569	0.012	195	92
J	charged-x-x-aliphatic	1.0	0.6056	0.673	0.015	94	60
K	every_fifth-navy	1.0	0.5645	0.555	0.017	177	118
L	vowels-x-x-first_five	1.0	0.5954	0.632	0.018	101	63
	every_fifth-navy	1.0	0.5645				

Table_22. Performance on out-of-sample test sets for different series of the artificial peptide datasets containing different inserted patterns. For disambiguation of the amino acid groups, see Table 6 and Table 21. BAC: balanced accuracy; ***PIP***: pattern insertion probability; ***POP***: pattern occurrence probability. The means and SDs are based on ***n***=20 tests.

The dataset-specific optimization of the number ***k*** of features led to another observation: although optimal ***k*** shows high variance, there is a negative correlation between ***k*** and the performance. This trend is logical, as both, low ***k*** and higher performance have a common cause: stronger patterns require fewer features for successful detection. From the optimization logic, direct causality between ***k*** and the performance is excluded from consideration. The most remarkable difference is between the series B and C. They demonstrate comparable performance achieved at very different optimal ***k*** values. Indeed, in series B, there is a pattern exactly corresponding to one of the features analyzed but diluted with random samples, while in series C, there is a persistent pattern, which must be replicated through other features. Thus, conducting optimization of ***k*** *in situ* also made the pipeline adaptable for scenarios when the putative patterns are not known in advance, which is the real world case.

4. DISCUSSION

4.1. Discussion overview

In this discussion, I first describe the innovations introduced in the SLiM predictor HH-MOTiF in comparison to similar existing software tools. Then, I explain, how the development of the positional data analyzer SLALOM facilitated, among other tasks, the performance comparison between different SLiM predictors. Next, I address the challenges in interpreting the performance metrics values that arise due to relatively small number of the experimentally verified SLiMs and high diversity in properties of those. Finally, I evaluate the successfulness of methods for detection of SLiMs of any type to assist in a *de novo* SLiM search.

4.2. The novelty in the SLiM searching pipeline of HH-MOTiF

4.2.1. Notes to the pipeline design

Although computational *de novo* SLiM searching is not a new field, there are some distinct innovations introduced in HH-MOTiF. These innovations include already known bioinformatics techniques extensively used for other purposes (e.g., for multiple sequence alignments or search for conserved domains) but not yet for SLiM search, as well as novel algorithms that can potentially find their application in other areas (e.g., the alignment recognition algorithm may be adapted for the more general problem of multiple sequence alignment). The proposed pipeline is able to outperform on average all other SLiM predictors available to date.

4.2.2. Conservation-based search for orthologs

The HH-MOTiF pipeline begins with the search for orthologs of all sequences in the input dataset. This by itself is usually not part of other SLiM predictors' pipelines, such as MEME-Suite (Bailey et al., 2009)(Bailey et al., 2015) and SLiMFinder (Edwards et al., 2007). This task is therefore left to the user. The DILIMOT web-server (Neduva and Russell, 2006) offers such a functionality by fetching the pre-compiled ortholog lists of several organisms (17, as of 12.07.2017). This, however, limits the number of orthologs that can be found, as well as prohibits any ortholog searches for organisms not in the list. On the other hand, pre-computation offers significant speed advantages. In addition, it opens the room for streamlining the usage of high-quality manual ortholog

annotations from different databases, which would show higher sensitivity than fully software-based ortholog detection procedures. However, such a half-automated protocol is currently not yet implemented in DILIMOT. HH-MOTiF uses reciprocal BLAST searches to detect orthologs. The BLAST searches against the NCBI `nr` database support virtually all known proteins from all organisms at the cost of the speed. The speed problem, however, is partially addressed by caching of ortholog search results.

Another aspect, which I noticed during the development of HH-MOTiF, is that only the orthologs in a certain similarity range are useful for considering the evolutionary conservation in the context of a SLiM search. The general tendency is that the conservation of SLiMs and their flanking regions is slightly higher than that of the remaining sequence (Chica et al., 2009). However, too close orthologs have too few mutations to be informative, while too distant ones have a high chance to have the SLiM moved along the sequence (e.g., the Dpb11/TopBP1 interaction motif in Slx4 proteins from different eukaryotes (Gritenaite et al., 2014)) or lost altogether. The level of conservation between orthologs varies heavily from one protein to the other, even for the same pair of organisms, like human and mouse (Makałowski et al., 1996). Therefore, I speculate that the specified cut-off levels based on coverage and identity are a more sensible solution than using orthologs from a predefined set of organisms regardless of the conservation levels.

4.2.3. Building HMM profiles from aligned orthologs

The identified orthologs are aligned and subsequently transformed to hidden Markov model profiles. Consequently, positions (columns) will have a score, which reflects the level of conservation. Thus, HH-MOTiF will give a lower (worse) score to weakly conserved columns at the profile-to-profile comparison step. This is different from other tools such as SLiMFinder, DILIMOT, whmm (Song and Gu, 2015), dhmm (Song et al., 2015), which implement the all-or-nothing decision: each residue either is considered as completely conserved or is fully ignored. Therefore, the HH-MOTiF approach considers evolutionary information in a much more elegant way and eliminates a need for an additional threshold to distinguish between conserved and non-conserved residues. In addition, operating with orthologs allows us to compare mutation patterns of the corresponding positions in a pair of regions from different query proteins. Regions with

similar mutation patterns get higher scores in `hhalign` alignments and therefore have a higher chance to be integrated into a common motif tree. Profile-to-profile HMM alignments have found their application in the search for remote homologs both, for protein (Deng and Cheng, 2014b) and DNA (Wheeler and Eddy, 2013) sequences. Furthermore, they were successfully used for the identification of protein domains (Zhang and Sun, 2012), as well as DNA sequence elements (Edlefsen and Liu, 2010). There is also a SLiM predictor, which cuts out SLiMs from surrounding domains (Horan et al., 2010). However, to my best knowledge, HH-MOTiF is the pioneering tool for *de novo* SLiM identification by profile-to-profile alignments of HMMs which are built from orthologs.

4.2.4. Separate masking of buried and ordered sequence regions

After HMMs are built, HH-MOTiF performs binary residue-wise masking of buried, globular regions as well as of overly ordered regions with subsequent resolving of the obtained mask. The ordered regions masking is implemented as in SLiMFinder. The resolving is also implemented in a way similar to SLiMFinder, although this step is not mentioned in the manuscript (Edwards et al., 2007), nor in the manual downloaded from the website of the author's group. The surface exposure-based masking is also not a new invention, as it is offered by, among others, DILIMOT. In this work, I show that such masking with NetSufpP (Petersen et al., 2009) indeed masks non-SLiM residues with reasonable selectivity, which was not explicitly shown before.

The exposed and disordered regions are partially related, as disordered residues, although they may occasionally become buried in some conformations the region adopts, will be accessible to the solvent for at least some time, and residues buried inside globular domains should remain ordered to maintain the conformation of the domain. In (Sheriff et al., 1985), it is shown that the surface exposure is related to the flexibility. The flexibility in turn correlates with the disorder, although these two terms should not be confused (Radivojac et al., 2004). In (Edwards and Palopoli, 2015) it is stated that only approx. 15% of SLiMs lie on the surface of globular domains, with the remaining 85% being predominantly in disordered regions. This can be the reason for Edwards *et al.* to implement only disordered region-, but not exposure masking in SLiMFinder. Nevertheless, in our work (Prytuliak et al., 2017), we showed that these two

types of masking have a quite different impact on the performance of both, SLiMFinder and HH-MOTiF, and therefore should be evaluated separately. Intriguingly, surface exposure masking, despite being selective per se, does not boost the final performance of either of the two tools,(see section 4.5. On filtering by sequence properties in *de novo* SLiM prediction for details).

4.2.5. Residue-wise accumulation of alignment statistics

After completing the pairwise HMM-HMM comparisons, HH-MOTiF performs the residue-wise accumulation of the scores. This is performed for each query sequence separately. This approach is different from that of other tools, which manipulate inter-query SLiM representations, in the form of either regular expressions (SLiMFinder, MEME, MotifHound (Kelil et al., 2014)) or suboptimal alignments (GLAM2 (Frith et al., 2008)), throughout their pipelines. The approach of HH-MOTiF, on the other hand, in that matter is similar to that of MFSPSSMpred (Fang et al., 2013), which assigns each residue its 'SLiM-likeness' based on a machine learning prediction. Unlike the latter, though, HH-MOTiF operates in the context of a small dataset and computes its 'SLiM-likeness' score not on the basis of similarity to a generalized feature profile of a collection of known SLiMs, but on the basis of higher than usual similarity to other proteins from the input dataset. Concerning this aspect, however, it is not possible to provide any evidence showing superiority or disadvantage of this approach, as it is not possible to construct a tool that would have only this part replaced with all other algorithm steps intact.

4.2.6. Hierarchical representation of SLiMs

The representation of SLiM candidates in HH-MOTiF is also unique. While other tools operate with regular expressions, profiles, and multiple sequence alignments, HH-MOTiF builds hierarchical structures, which are referred to as motif trees.

Each motif tree has exactly one motif root, which is aligned with all other motif instances – the so-called motif leaves. Motif leaves, however, are not strictly aligned with each other, which brings a hierarchy to the SLiM representation. In other tools, all the motif instances are always equal and maintain a kind of an 'all-to-all' alignment. The initial motivation for such a representation in HH-MOTiF was to mimic an evolutionary

process, where a motif would have an ‘ancestor’ and a number of ‘descendants’. In theory, if the evolution of a motif is divergent, the ancestor will be more similar to its descendants than the descendants are among each other. Unfortunately, motif evolution is hard to trace, as there is very little information on experimentally verified motifs of the same type in orthologous proteins throughout multiple species. If it is available, it is often not conclusive: e.g., in the ELM class LIG_WRPW_1 the instances in the orthologous proteins are too well conserved to support or reject the theory. Moreover, there are indications that the motif evolution is not divergent but rather convergent (Chemes et al., 2012). Furthermore, even if the evolution of a particular SLiM is strictly divergent, it is not guaranteed that a given dataset contains the ancestor instance and not only a subset of descendants. Taken together, a hierarchical motif structure does not reflect a mimic of the evolutionary process in contrast to my initial motivation.

Nevertheless, the hierarchical approach of HH-MOTiF demonstrated an unexpected advantage. As it was already discussed, the similarity between SLiMs extends to their flanking regions. The flanking regions, however, are not part of the SLiM, and consequently are not included in the database annotations. Therefore, they should not be reported in the output to the user; flanking regions that are reported are treated as false positives in all the tests I performed. SLiMFinder performs well in strictly pointing at a SLiMs’ core regions; however, because of the generally low information content in SLiMs, many potential correct predictions are rejected as non-significant leading to the subdued recall with default configuration of SLiMFinder. GLAM2, on the other hand, has high recall, but tends to predict very long SLiM candidates, which demonstrates its inability to separate SLiMs and associated flanking regions. This is also supported by high divergence between the residue-wise and the site-wise precision of the default configuration of GLAM2 (see Table 10). HH-MOTiF combines the best of these two worlds: flanking regions influence the score of individual root-leaf alignments, but are trimmed away in the output due to the HH-MOTiF algorithm which concentrates on the well-conserved SLiM core. Thus, the enhanced conservation in the SLiM neighborhood helps HH-MOTiF to collect the critical mass of positions which subsequently can be reduced to the returned core. An example from the processing of the ELM class TRG_LysEnd_APsAcLL_3 is shown in Figure 12. In this example, the two initial alignment hits count 11 and 10 residues; however, they are trimmed to a motif tree of length 8 and

therefore cover the target motif of length 7. The matches in flanking residues contribute to the score and subsequently to the higher ranking of the alignment hits, but are subsequently trimmed in the output.

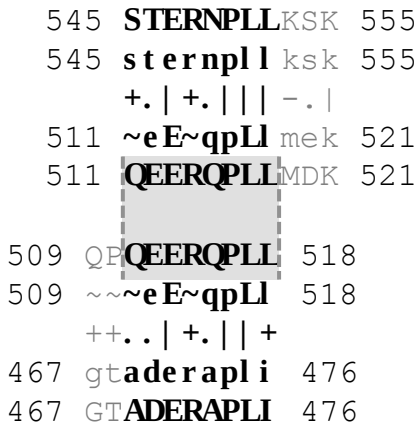


Figure 12. Two of the alignment hits found in the ELM class TRG_LysEnd_APsAcLL_3 dataset. They are forming a motif tree with the common stretch in the middle sequence (in the gray box) being the motif root and the bold marked stretches in the two other queries becoming the motif leaves; the grayed out residues are not part of the motif tree. The figure was reprinted from (Prytuliak et al., 2017) in agreement with the license no. 4200740316935. The copyright belongs to Oxford University Press.

However, the described concept of graceful handling of flanking residues will not work for those cases, where the flanking residues are conserved throughout the majority of the SLiM instances. In these cases, they will be considered as part of the SLiM. In such situations, the hard limit of five conserved positions, as implemented in SLiMFinder, presents a better solution.

4.2.7. Alignment recognition algorithm

The 'alignment recognition' algorithm was initially designed to control the maximal level of SLiM degeneration in motif trees, or to avoid situations where the motif root is relatively well aligned to each of the leaves, but the leaves are completely incompatible with and non-alignable to each other. A schematic example with four residue types is presented in the Figure 2.

The algorithm successfully tackles the described problem and in the end ensures reasonably high precision of HH-MOTiF (>55% site-wise) and is only outmatched by SLiMFinder with strict default settings. Alignment recognition has no direct analog in other SLiM predictors, because at this step, the pipeline of HH-MOTiF is cardinally

different: all other published tools work with multiple sequence alignments or profiles throughout the pipeline and have no need in combining pairwise alignments.

In addition, such an algorithm may come useful when dealing with the more general problem of integration of multiple sequence alignments. This problem is partially addressed in MAFFT v.7 through the options '`--merge`', '`--add`', '`--addprofile`', and '`--addfragments`' as well as through the separate mode '`mafft-profile`' (Katoh and Standley, 2013). Similar functionality is also implemented in PyMod (Bramucci et al., 2012). However, these tools address only the case of non-overlapping sequence sets in the input alignments (i.e., they can produce the four-sequence alignment A-B-C-D from the input A-B and C-D or A-B-C and D but not from A-B-C and A-C-D or A-B, A-C, and A-D). M-COFFEE (Wallace et al., 2006) (Moretti et al., 2007), which is part of T-COFFEE (Notredame et al., 2000), on the other hand, can combine information from several suboptimal alignments of the *same* sequences into a more reliable MSA. This means that it can take several different A-B-C-D alignments and produce another, better, A-B-C-D alignment; however, it still cannot integrate A-B-C and A-C-D into A-B-C-D. In addition, all these tools can neither perform further shortening of input local alignments (horizontal trimming) nor discard the worst matching sequences (vertical trimming). Therefore, the HH-MOTiF alignment recognition algorithm can potentially also fill the gap in existing approaches for integration of suboptimal local alignments. Nevertheless, to be applicable, it has to be first outsourced as a standalone application and additionally adjusted. This was not done in the scope of the current work.

4.2.8. Length penalties in the statistical model

Statistical evaluation of regex significance is a standard procedure during a SLiM search and is part of almost al predictors. It is common to calculate the probability of the regex to be assembled just by chance given the background frequencies of the amino acids with subsequent corrections for sequence and motif spaces. SLiMFinder implements perhaps the most comprehensive statistical model for SLiMs found *de novo* in a given dataset – SLiMChance. The statistical model of HH-MOTiF is similar to that of SLiMFinder, but has two key differences from SLiMChance.

First, it does not consider the motif space (explained in detail in Section 2.2.9. Regex generation and statistical evaluation). This may be the reason for HH-MOTiF's failure to reach the precision levels achieved with SLiMFinder. Nevertheless, the SLiMFinder's losses in recall are too large to be offset with the improvements in precision.

Second, it accounts for the query protein length differently. In short, in SLiMChance the effective number of independent tests (the power in the Šidák correction formula) grows linearly with the protein length $\boldsymbol{L}$, while in HH-MOTiF it grows as $\boldsymbol{L^{C-1}}$, where $\boldsymbol{C}$ is the number of conserved columns in the regex. The former is undoubtedly true for completely random sequences; however, it does not takes into account the biological reality, where certain amino acid patterns tend to occur more often than a pure calculation based on the background frequencies of individual amino acids would suggest. Such a pattern distribution skew may be observed because of better suitability of some patterns for alpha-helices (White and Jacobs, 1993) or because the protein is constructed from blocks duplicated in the course of evolution (Cahn et al., 2016). In the course of work on side projects with our collaboration partners, we for instance observed that false SLiM candidates can be abundant in the data sets of Ca-binding proteins, mainly due to the repeated EF-hand-like (Gifford et al., 2007) domains which are additionally surrounded by cysteine-rich regions to support the conformation through disulfide bonds (Kizawa et al., 2002)(Engel et al., 1987). These conserved domains are quite short and degenerate in comparison to other conserved domains (they are even sometimes called 'motifs', like in (Gifford et al., 2007)), and therefore they often pass the homology and domain filters of SLiM predictors. Moreover, proteins typically bear multiple instances of Ca-binding domains, some of which seem to be no longer functional but still have detectable sequence similarity. This leads to substantial pattern distribution skews in Ca-binding proteins, which in turn may result in extensive false positive predictions. Therefore, I speculate that a motif predictor requires stricter protein length discounting than a totally random model would imply, although the comprehensive and robust quantification of 'how much stricter' remained out of the scope of this work.

The HH-MOTiF model seems to be quite good in keeping the residue-wise precision at the same level across different average lengths of the input dataset. However, after

carrying out the same analysis for other tools, it is not clear if such smoothness is indeed achieved through the strict sequence length penalties. Although the optimized configurations of MEME and SLiMFinder, which both select top 5 hits scored by a length-linear penalty model, expectedly show a drop in precision upon rising query protein length, GLAM2 – which does not implement any explicit length penalties at all – demonstrates flat precision (see Table 23). Moreover, the high variance in performance between individual sets does not allow to draw firm conclusions from these data.

Average protein length		Residue-wise precision		
in the dataset	HH-MOTiF	SLiMFinder	GLAM2	MEME
>=1000 (n=28)	0.4245	0.2804	0.0764	0.0408
[500, 1000) (n=105)	0.4226	0.3242	0.0721	0.0614
<500 (n=43)	0.4141	0.5229	0.0735	0.0727

Table 23. Dependence of the average residue-wise precision on the average query protein length for different tools tested (optimized configurations used). The measurements were done on total n=176 ELM classes.

Intriguingly, the performance of SLiMFinder is maximized with the significance-based filtering being actually switched off and the output being limited to a fixed number of the best-ranked hits. For the optimized configuration I have used versus the default one, see Table 10; more SLiMFinder configurations are described in the publication (Prytuliak et al., 2017). I observed that the number of hits is also an important parameter for the HH-MOTiF performance; nevertheless, the best performance was achieved with the significance-based filtering of SLiM candidates turned on.

4.2.9. Contextual homology and conserved domain filtering

De novo SLiM searching is based on the detection of similar regions in query sequences. Therefore, the presence of similarities that arise for reasons other than common SLiMs obviously interferes with the search. Nevertheless, many SLiM predictors, among them MEME and GLAM2, do not address this issue, leaving the responsibility of handling conserved domains or redundant sequences, which are the most typical such non-SLiM similarities, to the user.

SLiMFinder addresses the problem by clustering the input sequences by similarity. This was also the initial approach implemented in HH-MOTiF. However, after extensive testing on different datasets, I found that clustering has some significant drawbacks originating from the fact that protein similarity measures are not Euclidian distances.

Generally, such situations require specifically adjusted clustering algorithms (Hathaway and Bezdek, 1994) and even then careful consideration of the logic is needed to avoid artifacts. The key question is what to do, when A is similar to both B and C, but B and C are not similar to each other (the problem is somewhat similar to the one with the motif trees – see Section 4.2.7. Alignment recognition alrorithm). The implications of this problem on SLiMFinder are already mentioned in Section 1.8.2. Conserved domains and larger homology regions. In addition, clustering algorithms I have tried (the SLiMFinders', Leaf (Bull et al., 2013), CD-HIT (Li and Godzik, 2006), as well as implementations of my own ideas) had a stability problem. The clusters often change significantly upon adding to or removing single sequences from a quite large dataset . Moreover, clustering results generally also depend on the order of the sequences in the dataset, and Leaf generates different results from run to run, even for the same input files due to initial random seeding. The order and randomization issues can be easily fixed through input sorting and constant seeding, respectively. However, the A-B-C and stability issues are not so easy to address.

In addition to the ambiguity of clustering, correcting for redundant sequences as a whole from the input dataset has another, unwanted implication: unique, non-similar regions will also be affected. This can happen, for example, for proteins that share a large conserved domain but are not related otherwise. To tackle this, DILIMOT offers a more elegant solution by masking out not the whole query sequences but only the similar regions. Such a solution in most cases avoids the problems of clustering or non-redundant subset generation, as well as handles gracefully partially homologous proteins. However, it still has one drawback, which I conceptually eliminated in HH-MOTiF.

DILIMOT homology masking is static, i.e. it is performed at the beginning of the motif searches without considering the context of actually identified motifs. The homologous regions, however, become a problem only, if there is an identified SLiM candidate that locates to more than one instance of the same region. SLiMs that locate to only one instance (which can happen, because the homologous regions must not be 100% identical and one can still harbor a motif that the other does not) should not be affected by homology filtering. Furthermore, it can be argued that if a SLiM does locate to

homologous regions, the highest-scored from all the affected SLiM instances should be retained for further analysis of the whole SLiM. However, if masking is performed before the search, there is no way to know, which of the homologous proteins will harbor the best SLiM representative. Finally, if there are overlapping homologous regions in more than two proteins (i.e., A is homologous to B and B is homologous to C), we come to the same problem of biased choice of representatives as with whole sequence clustering.

Therefore, to combat these issues, I implemented the contextual homology masking. It does not affect input sequences per se but triggers the SLiM candidate re-check if it contains pairs of instances located to the corresponding homologous regions. This re-check does not remove the individual leaves: if the tree has enough instances in non-homologous parts of the query proteins, it is retained and the user can see all the identified instances, including the homologous ones. The problem of the biased representatives choice was not fully tackled in the published version of HH-MOTiF. Namely, in the A-B-C case it was retaining the common homolog B, while more motif trees could be retained by discarding B and retaining non-homologous A and C instead. However, I fixed this issue in a newer version. This modification led to slight overall improvement in synthetic residue-wise F1 from 28.0% to 28.2% and in PC from 16.0% to 17.2%.

The provided data show that the homology masking plays an important role in the whole pipeline and affects the search results significantly. SLiMFinder does not process 68 out of 176 ELM classes at all, because the corresponding datasets are too homologous and therefore form less than three clusters of independent sequences. HH-MOTiF, however, was able to partially recover 19 of these 68 SLiMs. Moreover, 16 of them were identified with F1 score exceeding 0.28 (averaged result for all ELM classes), indicating the ability of HH-MOTiF to successfully overcome the dataset homology problem while identifying SLiMs, avoiding too stringent rejection of positives, as well as not allowing too much false positives to arise due to homology regions or conserved domains. The stringency of SLiMFinder, on the other hand, reduces substantially its resulting recall. Lack of homology filtering, however, can lead to low precision, which is what we observe for GLAM2.

4.2.10. Dynamic FASTA output format

The output format is an important part for a SLiM predictor, as it ensures easiness of understanding of the results for lab biologist not having much experience in programming or running console-based applications.

Displaying SLiM search results is conceptually not a hard task, when there is only one SLiM found in a relatively small dataset (3-5 proteins). The task gets more complicated, if there is a need to display multiple SLiMs in larger datasets. If the SLiM instances are overlapping, showing all of them together on the query sequences will lead to an incomprehensible meshwork. Therefore, some compromises are required.

To avoid overloading of the output page with information, DILIMOT displays one motif at a time on another page. Output motifs in MEME are not overlapping, so they are shown all together but marked with different colors. Output motif trees in HH-MOTiF are often overlapping; however, we did not want users to experience inconveniences associated with multiple pages. As the number of motif trees is theoretically unlimited, distinguishing them by colors is non-applicable. Therefore, we decided for a compromise of always showing only motif roots (which cannot overlap), while motif leaves become highlighted only upon selection of a tree. We implemented font stretching and appearing diagonal lines to make the selected tree instantly visible. To my best knowledge, HH-MOTiF is currently the only SLiM predictor with scrolling effects in the output page. The decision to show all the input sequences makes the motif positions very easy to see. On the down side, one needs a lot of scrolling in case of larger datasets and longer proteins sequences.

4.3. Advantages of statistical evaluations with SLALOM

SLALOM was developed to fill the niche in analyzing certain kinds of positional data, more specifically – the short motif predictions. The blueprint for SLALOM was the procedure described in (Song and Gu, 2015). However, their description lacks several important aspects. Namely, there is no information on handling overlaps in the prediction and in the ELM database, as well as no specification of the order of averaging. Moreover, the procedure is not available as standalone software.

The performance evaluation of other predictors being published was even less comprehensive. For example, in (Doğruel et al., 2008), (Edwards et al., 2007) and (Palopoli et al., 2015), only a binary classification of motifs as either “true or “false” predictions, and as “missed or “recovered” database motifs was provided. In (Kelil et al., 2014) the provided statistics are limited to residue-wise recall, which is referred to as ‘motif coverage’. Nevertheless, as shown throughout this thesis, the problem of a predictor’s performance evaluation is much more complex than just providing values of one metric for several datasets tested and therefore requires a separate statistical tool.

The statistical problems covered with SLALOM are generally not unique to the motif predictor cases and therefore were already mostly tackled in various software packages. For example, Bioconductor (Lawrence et al., 2013) and BEDtools (Quinlan, 2014) represent comprehensive solutions for genomics. With appropriate data pre-processing, they both can be used for estimating the performance of motif predictors. Nevertheless, they are from the concept site- or gene-oriented and as such do not provide the necessary flexibility to calculate residue-wise statistics. For example, they do not support gross residue counting. Moreover, they do not calculate performance metrics such as an F1 score. To expand potential areas of applicability, I additionally implemented time series processing in SLALOM. This functionality is also missing in the mentioned packages.

However, currently SLALOM remains a beta-version software.

4.4. SLiM heterogeneity as the challenge to statistical evaluation of predictors

Throughout this work, we often mention average values of different performance metrics across the ELM classes for different tools and configurations. However, we usually do not mention dispersion or significance values. This is caused by the fact that the values have different distributions across metrics, tools, and configurations. The only common feature of these distributions is the relatively large share of ELM classes with zero values (completely missed SLiMs), as one can see in Figure 13 A and C. Even if we remove all the zeros, the remaining values do not show a distinct pattern. Although the precision of HH-MOTiF (Figure 13 B) is distributed somewhat normally, this holds neither for the recall of HH-MOTiF, which exhibits almost perfect uniform distribution,

nor for the precision of SLiMFinder (Figure 13 D), which is clearly bimodal. The inability to determine the distribution type results in the impossibility of applying classical statistical tests such as ANOVA to evaluate the significance of the difference in performance between different tools or different configurations of the same tool.

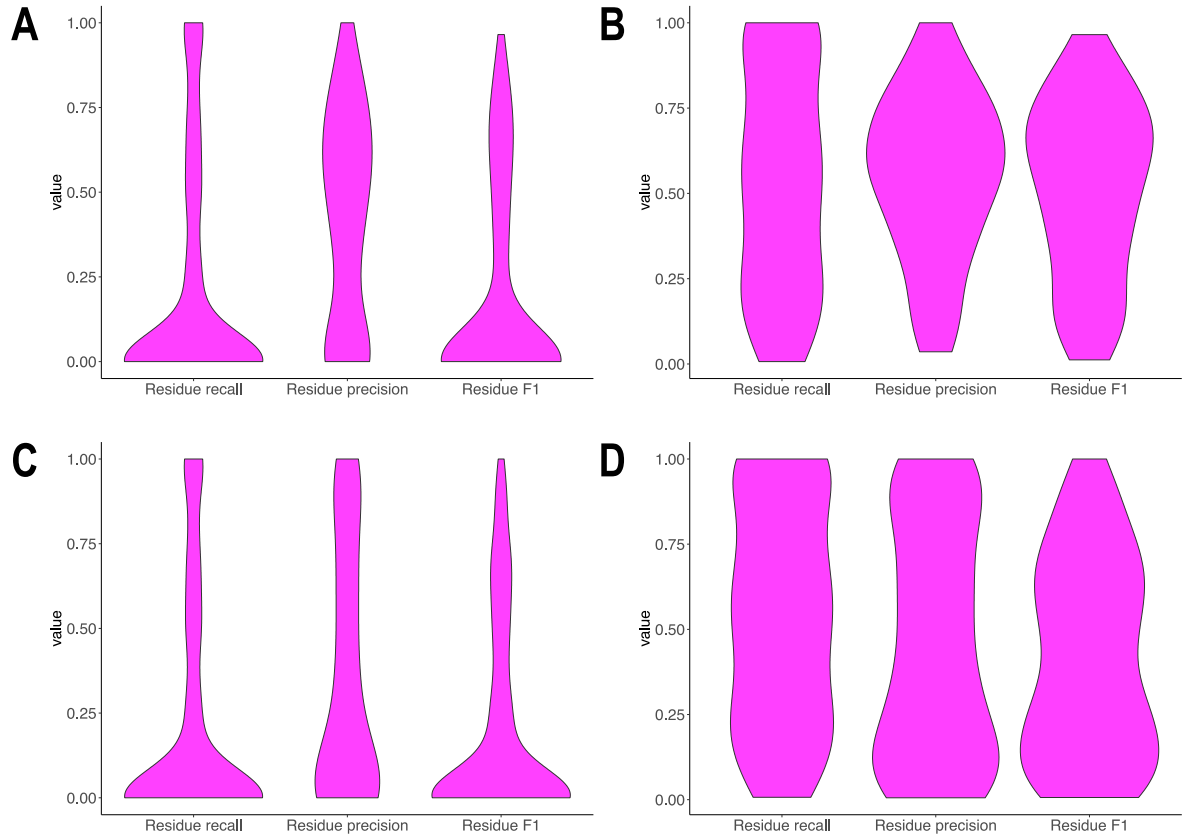


Figure 13. Violin plots, also known as smoothed bin histograms, showing the distribution of performance metric values among the analyzed ELM classes for the default configuration of HH-MOTiF (A, B) and the optimized configuration of SLiMFinder (C, D). All the 176 ELM classes are shown (A, C) or only those with non-zero performance (B, D).

The reason for the absence of a common distribution type for the performance values is the underlying heterogeneity of the ELM classes themselves. They do not represent 176 SLiM-containing datasets drawn at random from some large population. Instead, they represent a collection of datasets with different properties drawn from different sub-populations. Alternatively, we can say that the sample size in this case is not large enough to adequately capture the heterogeneity of the population.

We exclude zero and `nan` values for individual ELM classes from consideration on all the charts presented in this section, unless specified otherwise. Such a correction makes sense, as we want to quantify the dependence of prediction quality on different features.

The difference between zero and a positive value is on the other hand rather qualitative and not quantitative. We do not want zero values from SLiMs that cannot be predicted to distort the conclusions.

The factors that we identified as causing the heterogeneity are the following:

1. Diversity in the SLiM length
2. Diversity in the query protein length
3. Diversity in the dataset size
4. Diversity in the SLiM conservation
5. Diversity in the dataset intrinsic homology
6. Diversity in the relative position of the SLiM
7. Diversity in miscellaneous sequence properties (e.g. low complexity)

Those SLiMs, which are annotated in ELM vary in length from 2 to 23 residues. Longer SLiMs have theoretically better chances to be identified correctly. Indeed, GLAM2 shows a tendency to perform better on ELM classes with longer motifs (Figure 14 left). The same is true for MEME (Figure 14 right). This tendency is not random and can be explained by the general inability of these tools to trim SLiM candidates to the proper length: they can identify the shorter SLiMs but report too many flanking residues along with it. Logically, this affects precision but not recall. SLiMFinder and HH-MOTiF do not show this dependency; however, it was also observable in older versions of HH-MOTiF.

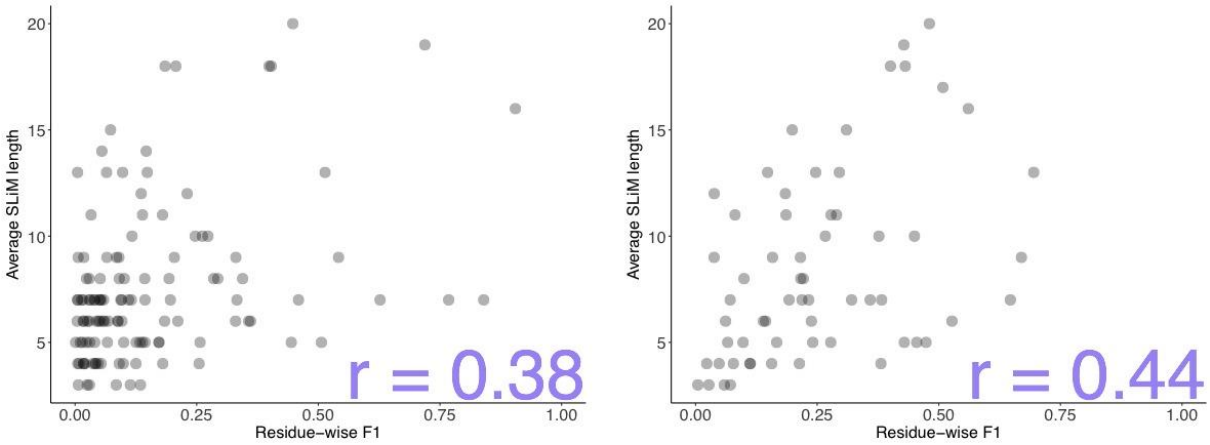


Figure 14. Scatter plot showing the correlation between SLiM length and the performance of GLAM2 (left) and MEME (right) with the optimized configuration. Each SLiM corresponds to an ELM class. Each point represents an ELM class; ***r*** is the correlation coefficient. Only ELM classes with non-zero performance are retained.

The conservation of SLiMs has a quite obvious influence on the performance. Indeed, we see that performance of both HH-MOTiF (Figure 15 left) and SLiMFinder (Figure 15 right) shows a correlation with the underlying SLiM conservation. Intriguingly, we could not detect this correlation for either MEME or GLAM2.

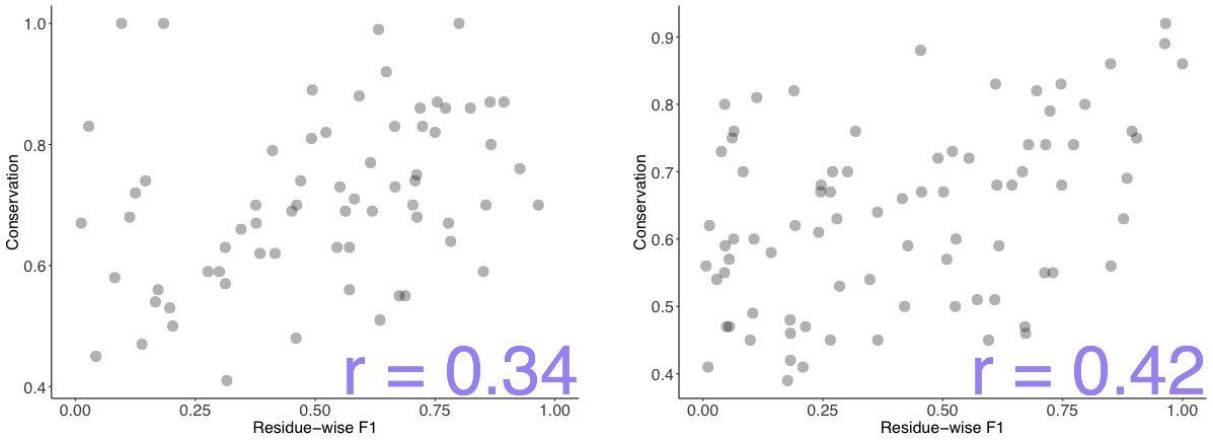


Figure 15. Scatter plots showing the correlation between SLiM conservation and predictability by HH-MOTiF (left) and SLiMFinder (right). Each SLiM corresponds to an ELM class. Only ELM classes with non-zero performance are retained. Each point represents an ELM class; ***r*** is the correlation coefficient.

The relative position may also make SLiMs harder or easier to find, as the terminal regions of the sequence may be influenced by various biases. MFSPWSSpred, for instance, cannot capture the termini, because it evaluates the SLiM-likeness of residues in the center of a sliding window. There are 14 ELM classes that represent exclusively N- or C-terminal motifs. As Table 24 shows, all the tools, except, perhaps, for MEME, show substantially different performances on these classes in comparison to the rest of the ELM database: HH-MOTiF and SLiMFinder find the terminal motifs better, while GLAM2 performs worse in these situations. In my opinion, GLAM2 indeed has a bias in the algorithm, as it finds the best SLiM candidates through moving the alignment from an initially random position; terminal positions in these situation have lower probability to be reached, although more detailed studies on the simulated annealing algorithm are needed to confirm or reject this hypothesis. However, the relatively low number of terminal motifs does not allow us to draw a firm conclusion from these data.

Terminal position of the	Residue-wise synthetic F1			
SLiM	HH-MOTiF	SLiMFinder	GLAM2	MEME
yes (n=14)	0.3754	0.3709	0.0616	0.1142
no (n=162)	0.2711	0.2483	0.1278	0.0934

Table 24. Dependence of the average residue-wise precision on the average query protein length for different tools tested (optimized configurations used). The measurements were done on total n=176 ELM classes.

Finally, a number of miscellaneous properties of the query sequences can impact the easiness of discovery of the SLiMs they harbor. The sequences, for example, may contain numerous low complexity regions (LCRs). When occurring in several input proteins, they will generate multiple false predictions. If a low complexity filter is implemented, it will mask those regions; however, this may hide true SLiMs that have low complexity nature. As one can see from Table 25, LCRs are more or less equally distributed between SLiM and non-SLiM residues. With the default settings of LCR filtering, the filter per se has even a negative impact on the performance, masking more SLiMs than unrelated LCRs. When LCRs are defined with the default window length, there are approx. 11% of ELM sites that have low complexity nature. Thus, there is a relatively large subgroup of SLiMs that require special handling from the perspective of a SLiM predictor. Similarly, input datasets can differ substantially in average surface exposure, sequence disorder, etc. The data in Table 26 show that approx. 7% of ELM sites are located in predicted buried regions. Moreover, the study from (Edwards and Palopoli, 2015) states that there are approx. 15% of SLiMs in ordered regions. Therefore, adding even a well-performing sequence property filter may result in the predictor becoming insensitive to a particular group of SLiMs, while gaining in performance on other groups. In addition, masking impacts the effective sequence length. If a large share of residues can be outright discarded as low complexity, buried or ordered, this leaves only few remaining positions for potential SLiM candidates. This makes the prediction easier. In such datasets, a predictor that implements the corresponding filters has an advantage over a predictor that does not. This, however, will not hold true for the datasets, where only few or no residues can be masked based on these features.

Measure	Low complexity window length		
	6	8 (default)	10
Share of non-SLiM residues covered ($LC_{nonSLiM}$), %	0.265	0.097	0.050
Share of SLiM residues covered (LC_{SLiM}), %	0.297	0.105	0.046
Share of ELM sites containing no low complexity residues, %	0.521	0.841	0.934
Share of ELM sites covered by LCRs to at least 50% (S_{LC50}), %	0.314	0.112	0.050

Table 25. Performance evaluation of an isolated low complexity filter on the ELM dataset. The default value of the window length in SLiMFinder and SEG (Wootton and Federhen, 1996) is 8.

Measure	RSA threshold	
	0.16	0.18
Share of non-SLiM residues buried ($B_{nonSLiM}$), %	0.183	0.202
Share of SLiM residues buried (B_{SLiM}), %	0.120	0.140
Share of ELM sites containing no buried residues, %	0.646	0.604
Share of ELM sites buried to at least 50% (S_{B50}), %	0.065	0.084

Table 26. Performance evaluation of an isolated surface exposure filter on the ELM dataset.

The influence of some other parameters has already been addressed in Results.

The list of factors causing the heterogeneity is probably still not complete.

The underlying heterogeneity in the input dataset properties leads to the heterogeneity in performance dataset-wise. A desired situation for comparison between different predictors would be a normally distributed performance. This could for instance have a mean of 0.4 and a standard deviation of 0.1. Unfortunately, as was already pointed out, the reality is quite different, with a substantial number of datasets containing no slim and performing exactly at the zero level and the rest of the values following a generally unknown distribution. This forced me to break the complete ELM dataset consisting of 176 separate datasets into subgroups and analyze the advantages and disadvantages of predictors in terms of certain specific SLiM and dataset properties, which was discussed in this thesis.

In addition, the heterogeneity can also explain, why the prediction of SLiM-likeness as a sequence property is much more challenging than for instance of secondary structure or surface exposure. The maximal reported balanced accuracy of SLiM-likeness prediction is currently only slightly over 60%. It has been reported with 61.0% for non-masked non-smoothed PSSM on the MoRF dataset in (Fang et al., 2013) and approx. 60% on the ELM data for the pipeline described in this thesis. Yet, it reaches almost 80% for surface exposure (Pugalenthi et al., 2012) and almost 90% for secondary structure (Feng et al., 2014). If the heterogeneity is removed and only one specific SLiM type is considered, a substantially higher accuracy (84%) can be reached (Kao et al., 2015).

4.5. On filtering by sequence properties in *de novo* SLiM prediction

De novo motif search relies on the identification of weak similarities between sequences in an input dataset. It does not identify SLiMs as such, but only enriched SLiMs in a given dataset. Therefore, it also does not directly rely on properties of a single sequence to operate. The same sequence region can be identified as SLiM in one dataset and as non-SLiM in another dataset. Even if some parts of the sequence can be outright masked solely on the basis of their properties, this will not affect the dataset performance, unless

there are similar stretches in a sufficient number of the other sequences in this dataset. This fact explains, why the discussed sequence property filters on low complexity, surface exposure and sequence disorder have relatively low – and sometimes even a negative – impact on the overall performance of predictors. For example, adding a sequence disorder filter decreases the residue-wise synthetic F1 of HH-MOTiF by 0.8 percentage points, but increases the performance of the default SLiMFinder configuration by 1.6 percentage points; the surface exposure filter on the other hand decreases the F1 by 0.5 and 4.5 points, respectively. The negative impact of the surface exposure filter can be to some extent explained by the insufficient performance of the filter itself. As Table 26 shows, the filter with the default threshold value of 0.16 masks only approx. 1.5 times more non-SLiM residues than of actual ELM sites. This means that the filter definitely has a certain selectivity per se; however, at the levels of ~20% recall and ~40% precision with the default HH-MOTiF configuration, the loss in recall will be too large to be consolidated by the gain in precision. Therefore, a slight drop in the resulting F1 score is expected. As the SLiMFinder precision is even higher, the drop is expectedly higher too. In fact, SLiMFinder with its ~65% precision is already more selective than the surface exposure filter. Therefore the latter can only 'dilute' the former, if combined.

Finally, it would be interesting to see, how filtering for the ultimate sequence property of the so-called SLiM-likeness would impact *de novo* SLiM prediction. It is interesting to note that the reported balanced accuracy 59.5% of my pipeline is consistent with 61.0% for non-smoothed non-masked reported in (Fang et al., 2014). Both approaches use randomized undersampling of the negative set, which leads to certain volatility in performance. I report the SD of 1.2 percentage points, while (Fang et al., 2014) reports only one value. Due to significant differences in the analyzed features, a direct comparison between the two approaches is not feasible. My approach does not require evolutionary information to work and is not designed to predict very terminal SLiMs. Nevertheless, the close performance values for distinct approaches indicate that it would be hard to achieve significantly higher performance. The balanced accuracy is further slightly increased to ~64% in MFSPSSmpred by combining the evolutionary conservation, surface exposure, and intrinsic disorder information with direct classification of PSSMs by SVM in one tool. Taken together, prediction of SLiM-likeness

itself is even less accurate than that of surface exposure and sequence disorder (Deng et al., 2015), I do not expect a significant improvement from such a filter. Nevertheless, the algorithmic novelties implemented in HH-MOTiF and discussed here already help to achieve previously unreachable levels in *de novo* SLiM search performance.

5. BIBLIOGRAPHY

Aldeghi, M., Heifetz, A., Bodkin, M.J., Knapp, S., and Biggin, P.C. (2016). Accurate calculation of the absolute free energy of binding for drug molecules. Chem. Sci. *7*, 207–218.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. (1990). Basic local alignment search tool. J. Mol. Biol. *215*, 403–410.

Altschul, S.F., Bundschuh, R., Olsen, R., and Hwa, T. (2001). The estimation of statistical parameters for local alignment score distributions. Nucleic Acids Res. *29*, 351–361.

Bailey, T.L., Williams, N., Misleh, C., and Li, W.W. (2006). MEME: discovering and analyzing DNA and protein sequence motifs. Nucleic Acids Res. *34*, W369-373.

Bailey, T.L., Boden, M., Buske, F.A., Frith, M., Grant, C.E., Clementi, L., Ren, J., Li, W.W., and Noble, W.S. (2009). MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res. *37*, W202-208.

Bailey, T.L., Johnson, J., Grant, C.E., and Noble, W.S. (2015). The MEME Suite. Nucleic Acids Res. *43*, W39-49.

Baştanlar, Y., and Özuysal, M. (2014). Introduction to Machine Learning. In miRNomics: MicroRNA Biology and Computational Analysis, (Humana Press, Totowa, NJ), pp. 105–128.

Baum, L.E., and Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. Ann. Math. Stat. *37*, 1554–1563.

Beauclair, G., Bridier-Nahmias, A., Zagury, J.-F., Saïb, A., and Zamborlini, A. (2015). JASSA: a comprehensive tool for prediction of SUMOylation sites and SIMs. Bioinforma. Oxf. Engl. *31*, 3483–3491.

Beck, J.R., and Shultz, E.K. (1986). The use of relative operating characteristic (ROC) curves in test performance evaluation. Arch. Pathol. Lab. Med. *110*, 13–20.

Benson, D.A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., and Sayers, E.W. (2013). GenBank. Nucleic Acids Res. *41*, D36–D42.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., and Bourne, P.E. (2000). The Protein Data Bank. Nucleic Acids Res. *28*, 235–242.

Blenner, M.A., Shur, O., Szilvay, G.R., Cropek, D.M., and Banta, S. (2010). Calcium-induced folding of a beta roll motif requires C-terminal entropic stabilization. J. Mol. Biol. *400*, 244–256.

Bonham-Carter, O., Steele, J., and Bastola, D. (2014). Alignment-free genetic sequence

comparisons: a review of recent approaches by word analysis. Brief. Bioinform. *15*, 890–905.

Bork, P., Dandekar, T., Diaz-Lazcoz, Y., Eisenhaber, F., Huynen, M., and Yuan, Y. (1998). Predicting function: from genes to genomes and back. J. Mol. Biol. *283*, 707–725.

Bramucci, E., Paiardini, A., Bossa, F., and Pascarella, S. (2012). PyMod: sequence similarity searches, multiple sequence-structure alignments, and homology modeling within PyMOL. BMC Bioinformatics *13 Suppl 4*, S2.

Breitenlechner, C., Engh, R.A., Huber, R., Kinzel, V., Bossemeyer, D., and Gassel, M. (2004). The Typically Disordered N-Terminus of PKA Can Fold as a Helix and Project the Myristoylation Site into Solution. Biochemistry (Mosc.) *43*, 7743–7749.

Brennan, R.G., and Matthews, B.W. (1989). The helix-turn-helix DNA binding motif. J. Biol. Chem. *264*, 1903–1906.

Bull, S.C., Muldoon, M.R., and Doig, A.J. (2013). Maximising the Size of Non-Redundant Protein Datasets Using Graph Theory. PLOS ONE *8*, e55484.

Cahn, J.K.B., Brinkmann-Chen, S., Buller, A.R., and Arnold, F.H. (2016). Artificial domain duplication replicates evolutionary history of ketol-acid reductoisomerases. Protein Sci. Publ. Protein Soc. *25*, 1241–1248.

Chemes, L.B., Glavina, J., Faivovich, J., de Prat-Gay, G., and Sánchez, I.E. (2012). Evolution of linear motifs within the papillomavirus E7 oncoprotein. J. Mol. Biol. *422*, 336–346.

Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T.J., Higgins, D.G., and Thompson, J.D. (2003). Multiple sequence alignment with the Clustal series of programs. Nucleic Acids Res. *31*, 3497–3500.

Chica, C., Diella, F., and Gibson, T.J. (2009). Evidence for the concerted evolution between short linear protein motifs and their flanking regions. PloS One *4*, e6052.

Colaert, N., Helsens, K., Martens, L., Vandekerckhove, J., and Gevaert, K. (2009). Improved visualization of protein consensus sequences by iceLogo. Nat. Methods *6*, 786–787.

Copp, S.M., Bogdanov, P., Debord, M., Singh, A., and Gwinn, E. (2014). Base motif recognition and design of DNA templates for fluorescent silver clusters by machine learning. Adv. Mater. Deerfield Beach Fla *26*, 5839–5845.

Crevenna, A.H., Blank, B., Maiser, A., Emin, D., Prescher, J., Beck, G., Kienzle, C., Bartnik, K., Habermann, B., Pakdel, M., et al. (2016). Secretory cargo sorting by Ca2+-dependent Cab45 oligomerization at the trans-Golgi network. J. Cell Biol. *213*, 305–314.

Crooks, G.E., Hon, G., Chandonia, J.-M., and Brenner, S.E. (2004). WebLogo: a sequence logo generator. Genome Res. *14*, 1188–1190.

Davey, N.E., Shields, D.C., and Edwards, R.J. (2006). SLiMDisc: short, linear motif discovery, correcting for common evolutionary descent. Nucleic Acids Res. *34*, 3546–3554.

Davey, N.E., Shields, D.C., and Edwards, R.J. (2009). Masking residues using context-specific evolutionary conservation significantly improves short linear motif discovery. Bioinformatics *25*, 443–450.

Davey, N.E., Haslam, N.J., Shields, D.C., and Edwards, R.J. (2011). SLiMSearch 2.0: biological context for short linear motifs in proteins. Nucleic Acids Res. *39*, W56-60.

Defenouillère, Q., Zhang, E., Namane, A., Mouaikel, J., Jacquier, A., and Fromont-Racine, M.

(2016). Rqc1 and Ltn1 Prevent C-terminal Alanine-Threonine Tail (CAT-tail)-induced Protein Aggregation by Efficient Recruitment of Cdc48 on Stalled 60S Subunits. J. Biol. Chem. *291*, 12245–12253.

Deng, X., and Cheng, J. (2014a). MSACompro: improving multiple protein sequence alignment by predicted structural features. Methods Mol. Biol. Clifton NJ *1079*, 273–283.

Deng, X., and Cheng, J. (2014b). Enhancing HMM-based protein profile-profile alignment with structural features and evolutionary coupling information. BMC Bioinformatics *15*, 252.

Deng, X., Gumm, J., Karki, S., Eickholt, J., and Cheng, J. (2015). An Overview of Practical Applications of Protein Disorder Prediction and Drive for Faster, More Accurate Predictions. Int. J. Mol. Sci. *16*, 15384–15404.

Deretic, D., Schmerl, S., Hargrave, P.A., Arendt, A., and McDowell, J.H. (1998). Regulation of sorting and post-Golgi trafficking of rhodopsin by its C-terminal sequence QVS(A)PA. Proc. Natl. Acad. Sci. U. S. A. *95*, 10620–10625.

Diella, F., Chabanis, S., Luck, K., Chica, C., Ramu, C., Nerlov, C., and Gibson, T.J. (2009). KEPE--a motif frequently superimposed on sumoylation sites in metazoan chromatin proteins and transcription factors. Bioinforma. Oxf. Engl. *25*, 1–5.

Dinkel, H., Van Roey, K., Michael, S., Kumar, M., Uyar, B., Altenberg, B., Milchevskaya, V., Schneider, M., Kühn, H., Behrendt, A., et al. (2016). ELM 2016--data update and new functionality of the eukaryotic linear motif resource. Nucleic Acids Res. *44*, D294-300.

Doğruel, M., Down, T.A., and Hubbard, T.J. (2008). NestedMICA as an ab initio protein motif discovery tool. BMC Bioinformatics *9*, 19.

Dosztányi, Z., Csizmók, V., Tompa, P., and Simon, I. (2005). The Pairwise Energy Content Estimated from Amino Acid Composition Discriminates between Folded and Intrinsically Unstructured Proteins. J. Mol. Biol. *347*, 827–839.

Dosztányi, Z., Mészáros, B., and Simon, I. (2009). ANCHOR: web server for predicting protein binding regions in disordered proteins. Bioinformatics *25*, 2745–2746.

Duncan, M.C., Costaguta, G., and Payne, G.S. (2003). Yeast epsin-related proteins required for Golgi-endosome traffic define a gamma-adaptin ear-binding motif. Nat. Cell Biol. *5*, 77–81.

Eddy, S.R. (1996). Hidden Markov models. Curr. Opin. Struct. Biol. *6*, 361–365.

Eddy, S.R. (2004). Where did the BLOSUM62 alignment score matrix come from? Nat. Biotechnol. *22*, 1035–1036.

Edlefsen, P.T., and Liu, J.S. (2010). Transposon identification using profile HMMs. BMC Genomics *11 Suppl 1*, S10.

Edwards, R., and Palopoli, N. (2015). Computational Prediction of Short Linear Motifs from Protein Sequences. In Computational Peptidology, P. Zhou, and J. Huang, eds. (Springer New York), pp. 89–141.

Edwards, R.J., Davey, N.E., and Shields, D.C. (2007). SLiMFinder: a probabilistic method for identifying over-represented, convergently evolved, short linear motifs in proteins. PloS One *2*, e967.

Endter, C., Kzhyshkowska, J., Stauber, R., and Dobner, T. (2001). SUMO-1 modification required for transformation by adenovirus type 5 early region 1B 55-kDa oncoprotein.

Proc. Natl. Acad. Sci. U. S. A. *98*, 11312–11317.

Engel, J., Taylor, W., Paulsson, M., Sage, H., and Hogan, B. (1987). Calcium binding domains and calcium-induced conformational transition of SPARC/BM-40/osteonectin, an extracellular glycoprotein expressed in mineralized and nonmineralized tissues. Biochemistry (Mosc.) *26*, 6958–6965.

Erdmann, F., Schäuble, N., Lang, S., Jung, M., Honigmann, A., Ahmad, M., Dudek, J., Benedix, J., Harsman, A., Kopp, A., et al. (2011). Interaction of calmodulin with Sec61α limits Ca2+ leakage from the endoplasmic reticulum. EMBO J. *30*, 17–31.

Fang, C., Noguchi, T., Tominaga, D., and Yamana, H. (2013). MFSPSSMpred: identifying short disorder-to-order binding regions in disordered proteins based on contextual local evolutionary conservation. BMC Bioinformatics *14*, 300.

Fang, C., Noguchi, T., and Yamana, H. (2014). Analysis of evolutionary conservation patterns and their influence on identifying protein functional sites. J. Bioinform. Comput. Biol. *12*, 1440003.

Feng, Y., Lin, H., and Luo, L. (2014). Prediction of protein secondary structure using feature selection and analysis approach. Acta Biotheor. *62*, 1–14.

Finn, R.D., Coggill, P., Eberhardt, R.Y., Eddy, S.R., Mistry, J., Mitchell, A.L., Potter, S.C., Punta, M., Qureshi, M., Sangrador-Vegas, A., et al. (2016). The Pfam protein families database: towards a more sustainable future. Nucleic Acids Res. *44*, D279–D285.

Foss, S.M., Li, H., Santos, M.S., Edwards, R.H., and Voglmaier, S.M. (2013). Multiple dileucine-like motifs direct VGLUT1 trafficking. J. Neurosci. Off. J. Soc. Neurosci. *33*, 10647–10660.

Frandsen, K.H., Rasmussen, K.K., Jensen, M.R., Hammer, K., Pedersen, M., Poulsen, J.-C.N., Arleth, L., and Lo Leggio, L. (2013). Binding of the N-Terminal Domain of the Lactococcal Bacteriophage TP901-1 CI Repressor to Its Target DNA: A Crystallography, Small Angle Scattering, and Nuclear Magnetic Resonance Study. Biochemistry (Mosc.) *52*, 6892–6904.

Frenkel-Morgenstern, M., and Valencia, A. (2012). Novel domain combinations in proteins encoded by chimeric transcripts. Bioinforma. Oxf. Engl. *28*, i67-74.

Frith, M.C., Saunders, N.F.W., Kobe, B., and Bailey, T.L. (2008). Discovering sequence motifs with arbitrary insertions and deletions. PLoS Comput. Biol. *4*, e1000071.

Gan, X.Q., Wang, J.Y., Yang, Q.H., Li, Z., Liu, F., Pei, G., and Li, L. (2000). Interaction between the conserved region in the C-terminal domain of GRK2 and rhodopsin is necessary for GRK2 to catalyze receptor phosphorylation. J. Biol. Chem. *275*, 8469–8474.

Gasser, A., Ho, T.S.-Y., Cheng, X., Chang, K.-J., Waxman, S.G., Rasband, M.N., and Dib-Hajj, S.D. (2012). An AnkyrinG-Binding Motif Is Necessary and Sufficient for Targeting $Na_v1.6$ Sodium Channels to Axon Initial Segments and Nodes of Ranvier. J. Neurosci. *32*, 7232–7243.

Gaur, R.K. (2014). Amino acid frequency distribution among eukaryotic proteins. IIOAB J. *5*, 6–11.

Gelman, A., Meng, X.-L., and Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. Stat. Sin. *6*, 733–807.

Gibson, T.J., Dinkel, H., Van Roey, K., and Diella, F. (2015). Experimental detection of

short regulatory motifs in eukaryotic proteins: tips for good practice as well as for bad. Cell Commun. Signal. CCS *13*, 42.

Gifford, J.L., Walsh, M.P., and Vogel, H.J. (2007). Structures and metal-ion-binding properties of the Ca2+-binding helix-loop-helix EF-hand motifs. Biochem. J. *405*, 199–221.

Gorelik, M., and Davidson, A.R. (2012). Distinct Peptide Binding Specificities of Src Homology 3 (SH3) Protein Domains Can Be Determined by Modulation of Local Energetics across the Binding Interface. J. Biol. Chem. *287*, 9168–9177.

Gritenaite, D., Princz, L.N., Szakal, B., Bantele, S.C.S., Wendeler, L., Schilbach, S., Habermann, B.H., Matos, J., Lisby, M., Branzei, D., et al. (2014). A cell cycle-regulated Slx4–Dpb11 complex promotes the resolution of DNA repair intermediates linked to stalled replication. Genes Dev. *28*, 1604–1619.

Hanley, J.A., and McNeil, B.J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology *143*, 29–36.

Hathaway, R.J., and Bezdek, J.C. (1994). Nerf c-means: Non-Euclidean relational fuzzy clustering. Pattern Recognit. *27*, 429–437.

Hess, M., Keul, F., Goesele, M., and Hamacher, K. (2016). Addressing inaccuracies in BLOSUM computation improves homology search performance. BMC Bioinformatics *17*, 189.

Hietakangas, V., Anckar, J., Blomster, H.A., Fujimoto, M., Palvimo, J.J., Nakai, A., and Sistonen, L. (2006). PDSM, a motif for phosphorylation-dependent SUMO modification. Proc. Natl. Acad. Sci. U. S. A. *103*, 45–50.

Horan, K., Shelton, C.R., and Girke, T. (2010). Predicting conserved protein motifs with Sub-HMMs. BMC Bioinformatics *11*, 205.

Hornbeck, P.V., Zhang, B., Murray, B., Kornhauser, J.M., Latham, V., and Skrzypek, E. (2015). PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. Nucleic Acids Res. *43*, D512-520.

Huang, C.-H., Su, M.-G., Kao, H.-J., Jhong, J.-H., Weng, S.-L., and Lee, T.-Y. (2016). UbiSite: incorporating two-layered machine learning method with substrate motifs to predict ubiquitin-conjugation site on lysines. BMC Syst. Biol. *10 Suppl 1*, 6.

Hudson, A.M., and Cooley, L. (2013). Phylogenetic, Structural and Functional Relationships between WD-and Kelch-Repeat Proteins (Landes Bioscience).

Jacomin, A.-C., Samavedam, S., Promponas, V., and Nezis, I.P. (2016). iLIR database: A web resource for LIR motif-containing proteins in eukaryotes. Autophagy *12*, 1945–1953.

Kao, H.-J., Huang, C.-H., Bretaña, N.A., Lu, C.-T., Huang, K.-Y., Weng, S.-L., and Lee, T.-Y. (2015). A two-layered machine learning method to identify protein O-GlcNAcylation sites with O-GlcNAc transferase substrate motifs. BMC Bioinformatics *16 Suppl 18*, S10.

Karaçali, B. (2012). Hierarchical motif vectors for prediction of functional sites in amino acid sequences using quasi-supervised learning. IEEE/ACM Trans. Comput. Biol. Bioinform. *9*, 1432–1441.

Katoh, K., and Standley, D.M. (2013). MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol. Biol. Evol. *30*, 772–780.

Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. *30*, 3059–3066.

Kelil, A., Dubreuil, B., Levy, E.D., and Michnick, S.W. (2014). Fast and accurate discovery of degenerate linear motifs in protein sequences. PloS One *9*, e106081.

Kim, J., Pramanik, S., and Chung, M.J. (1994). Multiple sequence alignment using simulated annealing. Comput. Appl. Biosci. CABIOS *10*, 419–426.

Kinjo, A., and Nakamura, H. (2013). Functional Structural Motifs for Protein–Ligand, Protein–Protein, and Protein–Nucleic Acid Interactions and their Connection to Supersecondary Structures. In Protein Supersecondary Structures, A.E. Kister, ed. (Humana Press), pp. 295–315.

Kizawa, K., Troxler, H., Kleinert, P., Inoue, T., Toyoda, M., Morohashi, M., and Heizmann, C.W. (2002). Characterization of the cysteine-rich calcium-binding S100A3 protein from human hair cuticles. Biochem. Biophys. Res. Commun. *299*, 857–862.

Kleene, S.C., Shannon, C.E., and McCarthy, J. (1956). Representation of Events in Nerve Nets and Finite Automata. Automata Studies. In Automata Studies, (Princeton University Press), pp. 3–42.

Kumaran Nair, S.S., Subba Reddy, N.V., and Hareesha, K.S. (2012). Machine learning study of classifiers trained with biophysiochemical properties of amino acids to predict fibril forming Peptide motifs. Protein Pept. Lett. *19*, 917–923.

Laskowski, R.A., Luscombe, N.M., Swindells, M.B., and Thornton, J.M. (1996). Protein clefts in molecular recognition and function. Protein Sci. Publ. Protein Soc. *5*, 2438–2452.

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., Morgan, M.T., and Carey, V.J. (2013). Software for Computing and Annotating Genomic Ranges. PLOS Comput. Biol. *9*, e1003118.

Lee, T.-Y., Lin, Z.-Q., Hsieh, S.-J., Bretaña, N.A., and Lu, C.-T. (2011). Exploiting maximal dependence decomposition to identify conserved motifs from a group of aligned signal sequences. Bioinforma. Oxf. Engl. *27*, 1780–1787.

Li, W., and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinforma. Oxf. Engl. *22*, 1658–1659.

Lin, T., Murphy, R.F., and Bar-Joseph, Z. (2011). Discriminative motif finding for predicting protein subcellular localization. IEEE/ACM Trans. Comput. Biol. Bioinform. *8*, 441–451.

Liu, B.A., and Nash, P.D. (2012). Evolution of SH2 domains and phosphotyrosine signalling networks. Philos. Trans. R. Soc. Lond. B Biol. Sci. *367*, 2556–2573.

Liu, Z., Gong, Z., Dong, X., and Tang, C. (2016). Transient protein-protein interactions visualized by solution NMR. Biochim. Biophys. Acta *1864*, 115–122.

Lui, W.W.Y., Collins, B.M., Hirst, J., Motley, A., Millar, C., Schu, P., Owen, D.J., and Robinson, M.S. (2003). Binding partners for the COOH-terminal appendage domains of the GGAs and gamma-adaptin. Mol. Biol. Cell *14*, 2385–2398.

Makałowski, W., Zhang, J., and Boguski, M.S. (1996). Comparative analysis of 1196 orthologous mouse and human full-length mRNA and protein sequences. Genome Res. *6*, 846–857.

Malhotra, S., and Sowdhamini, R. (2015). Collation and analyses of DNA-binding protein domain families from sequence and structural databanks. Mol. Biosyst. *11*, 1110–1118.

Marchler-Bauer, A., Derbyshire, M.K., Gonzales, N.R., Lu, S., Chitsaz, F., Geer, L.Y., Geer, R.C., He, J., Gwadz, M., Hurwitz, D.I., et al. (2015). CDD: NCBI's conserved domain database. Nucleic Acids Res. *43*, D222–D226.

Mattera, R., Ritter, B., Sidhu, S.S., McPherson, P.S., and Bonifacino, J.S. (2004). Definition of the consensus motif recognized by gamma-adaptin ear domains. J. Biol. Chem. *279*, 8018–8028.

Meier, A., and Söding, J. (2015). Context similarity scoring improves protein sequence alignments in the midnight zone. Bioinforma. Oxf. Engl. *31*, 674–681.

Mi, T., Merlin, J.C., Deverasetty, S., Gryk, M.R., Bill, T.J., Brooks, A.W., Lee, L.Y., Rathnayake, V., Ross, C.A., Sargeant, D.P., et al. (2012). Minimotif Miner 3.0: database expansion and significantly improved reduction of false-positive predictions from consensus sequences. Nucleic Acids Res. *40*, D252-260.

Milewski, M.I., Mickle, J.E., Forrest, J.K., Stafford, D.M., Moyer, B.D., Cheng, J., Guggino, W.B., Stanton, B.A., and Cutting, G.R. (2001). A PDZ-binding motif is essential but not sufficient to localize the C terminus of CFTR to the apical membrane. J. Cell Sci. *114*, 719–726.

Mills, L.J., and Pearson, W.R. (2013). Adjusting scoring matrices to correct overextended alignments. Bioinforma. Oxf. Engl. *29*, 3007–3013.

Mitchell, A., Chang, H.-Y., Daugherty, L., Fraser, M., Hunter, S., Lopez, R., McAnulla, C., McMenamin, C., Nuka, G., Pesseat, S., et al. (2015). The InterPro protein families database: the classification resource after 15 years. Nucleic Acids Res. *43*, D213–D221.

Mohamed, R., Degac, J., and Helms, V. (2015). Composition of Overlapping Protein-Protein and Protein-Ligand Interfaces. PLOS ONE *10*, e0140965.

Moretti, S., Armougom, F., Wallace, I.M., Higgins, D.G., Jongeneel, C.V., and Notredame, C. (2007). The M-Coffee web server: a meta-method for computing multiple sequence alignments by combining alternative alignment methods. Nucleic Acids Res. *35*, W645-648.

Necci, M., Piovesan, D., and Tosatto, S.C.E. (2016). Large-scale analysis of intrinsic disorder flavors and associated functions in the protein sequence universe. Protein Sci. *25*, 2164–2174.

Neduva, V., and Russell, R.B. (2006). DILIMOT: discovery of linear motifs in proteins. Nucleic Acids Res. *34*, W350–W355.

Neduva, V., Linding, R., Su-Angrand, I., Stark, A., Masi, F. de, Gibson, T.J., Lewis, J., Serrano, L., and Russell, R.B. (2005). Systematic Discovery of New Recognition Peptides Mediating Protein Interaction Networks. PLOS Biol. *3*, e405.

Neupert, W., and Herrmann, J.M. (2007). Translocation of proteins into mitochondria. Annu. Rev. Biochem. *76*, 723–749.

Neuwald, A.F., Liu, J.S., and Lawrence, C.E. (1995). Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein Sci. Publ. Protein Soc. *4*, 1618–1632.

Nielsen, M.B., Birkeland, M.S., Hansen, M.B., Knardahl, S., and Heir, T. (2017). Victimization from workplace bullying after a traumatic event: time-lagged relationships

with symptoms of posttraumatic stress. Int. Arch. Occup. Environ. Health *90*, 411–421.

Notredame, C., Higgins, D.G., and Heringa, J. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. *302*, 205–217.

Nunes, B., Natário, I., and Carvalho, M.L. (2011). Time series methods for obtaining excess mortality attributable to influenza epidemics. Stat. Methods Med. Res. *20*, 331–345.

Obenauer, J.C., Cantley, L.C., and Yaffe, M.B. (2003). Scansite 2.0: Proteome-wide prediction of cell signaling interactions using short sequence motifs. Nucleic Acids Res. *31*, 3635–3641.

Oeste, C.L., Pinar, M., Schink, K.O., Martínez-Turrión, J., Stenmark, H., Peñalva, M.A., and Pérez-Sala, D. (2014). An isoprenylation and palmitoylation motif promotes intraluminal vesicle delivery of proteins in cells from distant species. PloS One *9*, e107190.

O'Neal, K.D., Yu-Lee, L.Y., and Shearer, W.T. (1995). The proline-rich motif is necessary but not sufficient for prolactin receptor signal transduction. Ann. N. Y. Acad. Sci. *766*, 282–284.

Palopoli, N., Lythgow, K.T., and Edwards, R.J. (2015). QSLiMFinder: improved short linear motif prediction using specific query protein data. Bioinformatics *31*, 2284–2293.

Petersen, B., Petersen, T.N., Andersen, P., Nielsen, M., and Lundegaard, C. (2009). A generic method for assignment of reliability scores applied to solvent accessibility predictions. BMC Struct. Biol. *9*, 51.

Pierce, M.M., Raman, C.S., and Nall, B.T. (1999). Isothermal titration calorimetry of protein-protein interactions. Methods San Diego Calif *19*, 213–221.

Poirot, O., Suhre, K., Abergel, C., O'Toole, E., and Notredame, C. (2004). 3DCoffee@igs: a web server for combining sequences and structures into a multiple sequence alignment. Nucleic Acids Res. *32*, W37-40.

Pruitt, K., Brown, G., Tatusova, T., and Maglott, D. (2012). The Reference Sequence (RefSeq) Database (National Center for Biotechnology Information (US)).

Prytuliak, R., Volkmer, M., Meier, M., and Habermann, B.H. (2017). HH-MOTiF: de novo detection of short linear motifs in proteins by Hidden Markov Model comparisons. Nucleic Acids Res.

Pugalenthi, G., Kandaswamy, K.K., Chou, K.-C., Vivekanandan, S., and Kolatkar, P. (2012). RSARF: prediction of residue solvent accessibility from protein sequence using random forest method. Protein Pept. Lett. *19*, 50–56.

Quinlan, A.R. (2014). BEDTools: The Swiss-Army Tool for Genome Feature Analysis. Curr. Protoc. Bioinforma. *47*, 11.12.1-34.

Radivojac, P., Obradovic, Z., Smith, D.K., Zhu, G., Vucetic, S., Brown, C.J., Lawson, J.D., and Dunker, A.K. (2004). Protein flexibility and intrinsic disorder. Protein Sci. *13*, 71–80.

Remmert, M., Biegert, A., Hauser, A., and Söding, J. (2011). HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat. Methods *9*, 173–175.

Ren, S., Yang, G., He, Y., Wang, Y., Li, Y., and Chen, Z. (2008). The conservation pattern of short linear motifs is highly correlated with the function of interacting protein domains. BMC Genomics *9*, 452.

Renart, J., Reiser, J., and Stark, G.R. (1979). Transfer of proteins from gels to diazobenzyloxymethyl-paper and detection with antisera: a method for studying antibody specificity and antigen structure. Proc. Natl. Acad. Sci. U. S. A. *76*, 3116–3120.

Rigoutsos, I., and Floratos, A. (1998). Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. Bioinforma. Oxf. Engl. *14*, 55–67.

Rose, P.W., Prlić, A., Bi, C., Bluhm, W.F., Christie, C.H., Dutta, S., Green, R.K., Goodsell, D.S., Westbrook, J.D., Woo, J., et al. (2015). The RCSB Protein Data Bank: views of structural biology for basic and applied research and education. Nucleic Acids Res. *43*, D345-356.

Schneider, T.D., and Stephens, R.M. (1990). Sequence logos: a new way to display consensus sequences. Nucleic Acids Res. *18*, 6097–6100.

Schneider, T.D., Stormo, G.D., Gold, L., and Ehrenfeucht, A. (1986). Information content of binding sites on nucleotide sequences. J. Mol. Biol. *188*, 415–431.

Schölkopf, B., and Smola, A.J. (2001). Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond (Cambridge, MA: MIT Press).

Schultz, J., Milpetz, F., Bork, P., and Ponting, C.P. (1998). SMART, a simple modular architecture research tool: Identification of signaling domains. Proc. Natl. Acad. Sci. *95*, 5857–5864.

Sheldon, K.L., Gurnev, P.A., Bezrukov, S.M., and Sackett, D.L. (2015). Tubulin Tail Sequences and Post-translational Modifications Regulate Closure of Mitochondrial Voltage-dependent Anion Channel (VDAC). J. Biol. Chem. *290*, 26784–26789.

Shen, L., Shao, N.-Y., Liu, X., Maze, I., Feng, J., and Nestler, E.J. (2013). diffReps: Detecting Differential Chromatin Modification Sites from ChIP-seq Data with Biological Replicates. PLOS ONE *8*, e65598.

Sheriff, S., Hendrickson, W.A., Stenkamp, R.E., Sieker, L.C., and Jensen, L.H. (1985). Influence of solvent accessibility and intermolecular contacts on atomic mobilities in hemerythrins. Proc. Natl. Acad. Sci. U. S. A. *82*, 1104–1107.

Siebert, M., and Söding, J. (2016). Bayesian Markov models consistently outperform PWMs at predicting motifs in nucleotide sequences. Nucleic Acids Res. *44*, 6055–6069.

Sievers, F., Wilm, A., Dineen, D., Gibson, T.J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., et al. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol. Syst. Biol. *7*, 539.

Söding, J. (2005). Protein homology detection by HMM-HMM comparison. Bioinforma. Oxf. Engl. *21*, 951–960.

Song, T., and Gu, H. (2015). Discovering short linear protein motif based on selective training of profile hidden Markov models. J. Theor. Biol. *377*, 75–84.

Song, T., Bu, X., and Gu, H. (2015). Combining intrinsic disorder prediction and augmented training of hidden Markov models improves discriminative motif discovery. Chem. Phys. Lett. *634*, 243–248.

Stormo, G.D., Schneider, T.D., Gold, L., and Ehrenfeucht, A. (1982). Use of the "Perceptron" algorithm to distinguish translational initiation sites in E. coli. Nucleic Acids Res. *10*, 2997–3011.

Su, G., Morris, J.H., Demchak, B., and Bader, G.D. (2014). Biological network exploration with Cytoscape 3. Curr. Protoc. Bioinforma. *47*, 8.13.1-24.

Suvorova, I.A., Korostelev, Y.D., and Gelfand, M.S. (2015). GntR Family of Bacterial Transcription Factors and Their DNA Binding Motifs: Structure, Positioning and Co-Evolution. PloS One *10*, e0132618.

Szklarczyk, D., Franceschini, A., Wyder, S., Forslund, K., Heller, D., Huerta-Cepas, J., Simonovic, M., Roth, A., Santos, A., Tsafou, K.P., et al. (2015). STRING v10: protein-protein interaction networks, integrated over the tree of life. Nucleic Acids Res. *43*, D447-452.

Taskesen, E., Hoogeboezem, R., Delwel, R., and Reinders, M.J. (2013). Hypergeometric analysis of tiling-array and sequence data: detection and interpretation of peaks. Adv. Appl. Bioinforma. Chem. *2013*, 55–62.

Taylor, S.C., and Posch, A. (2014). The design of a quantitative western blot experiment. BioMed Res. Int. *2014*, 361590.

The UniProt Consortium (2017). UniProt: the universal protein knowledgebase. Nucleic Acids Res. *45*, D158–D169.

Topham, C.M., and Smith, J.C. (2015). Tri-peptide reference structures for the calculation of relative solvent accessible surface area in protein amino acid residues. Comput. Biol. Chem. *54*, 33–43.

Turnbull, A.P., and Emsley, P. (2013). Studying protein-ligand interactions using X-ray crystallography. Methods Mol. Biol. Clifton NJ *1008*, 457–477.

Vallon, M., Aubele, P., Janssen, K.-P., and Essler, M. (2012). Thrombin-induced shedding of tumour endothelial marker 5 and exposure of its RGD motif are regulated by cell-surface protein disulfide-isomerase. Biochem. J. *441*, 937–944.

Vyas, J., Nowling, R.J., Meusburger, T., Sargeant, D., Kadaveru, K., Gryk, M.R., Kundeti, V., Rajasekaran, S., and Schiller, M.R. (2010). MimoSA: a system for minimotif annotation. BMC Bioinformatics *11*, 328.

Wallace, I.M., O'Sullivan, O., Higgins, D.G., and Notredame, C. (2006). M-Coffee: combining multiple sequence alignment methods with T-Coffee. Nucleic Acids Res. *34*, 1692–1699.

Weatheritt, R.J., Jehl, P., Dinkel, H., and Gibson, T.J. (2012). iELM--a web server to explore short linear motif-mediated interactions. Nucleic Acids Res. *40*, W364-369.

Wheeler, T.J., and Eddy, S.R. (2013). nhmmer: DNA homology search with profile HMMs. Bioinforma. Oxf. Engl. *29*, 2487–2489.

White, S.H., and Jacobs, R.E. (1993). The evolution of proteins from random amino acid sequences. I. Evidence from the lengthwise distribution of amino acids in modern protein sequences. J. Mol. Evol. *36*, 79–95.

Wootton, J.C., and Federhen, S. (1996). Analysis of compositionally biased regions in sequence databases. Methods Enzymol. *266*, 554–571.

Wright, S.P. (1992). Adjusted P-Values for Simultaneous Inference. Biometrics *48*, 1005–1013.

Xu, X., Zhai, Y., Sun, F., Lou, Z., Su, D., Xu, Y., Zhang, R., Joachimiak, A., Zhang, X.C., Bartlam, M., et al. (2006). New antiviral target revealed by the hexameric structure of mouse hepatitis virus nonstructural protein nsp15. J. Virol. *80*, 7909–7917.

Zhang, Y., and Sun, Y. (2012). MetaDomain: a profile HMM-based protein domain

classification tool for short sequences. Pac. Symp. Biocomput. Pac. Symp. Biocomput. 271–282.