UNIVERSITY OF Southampton
School of Biological Sciences

SLiMSearch: Short Linear Motif Search Tool

Richard J. Edwards © 2008-2010.

Contents

Figures

Tables

1. Introduction

This manual gives an overview of SLiMSearch, a utility for searching biological sequences for Short, Linear Motifs (SLiMs). SLiMSearch is a reincarnation of an older program, PRESTO (Edwards 2006), rebuilt on the core architecture of SLiMFinder (Edwards *et al.* 2007) and contains much of the same functionality. The manuals for SLiMFinder and PRESTO, available at the website (http://www.personal.southampton.ac.uk/re1u06/software/), might therefore also be useful.

SLiMSearch is designed as a standalone application but additional python modules will need to be present for it to work. For full functionality, additional utilities must be installed; some of these require licenses and/or permissions of the authors. This document should describe these features but if anything is missing or needs clarification, please contact me. The fundamentals are covered in Chapter 2, Fundamentals, including input and output details. Later sections give more details on how the methods work and statistics are generated. General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to SLiMSearch can be found in the distributed readme.txt and readme.html files

Like the software itself, this manual is a 'work in progress' to some degree. If the version you are now reading does not make sense, then it may be worth checking the website to see if a more recent version is available, as indicated by the Version section of the manual. Check the readme on the website for up-to-date options etc. In particular, default values for options are subject to change and should be checked in the readme or by running the "help" command.

Rich Edwards, 2010.

1.1. Version

This manual is designed to accompany SLiMSearch version 1.5.

The manual was last edited on 04 June 2010.

1.2. Using this Manual

As much as possible, I shall try to make a clear distinction between explanatory text (this) and text to be typed at the command-prompt etc. Command prompt text will be `written in Courier New` to make the distinction clearer. Program options, also called 'command-line parameters', will be **`written in bold Courier New`** (and coloured **`red`** for fixed portions or **`dark`** for user-defined portions, such as file names etc.). Command-line examples will be given in *`italicised Courier New`*. Optional parameters will (if I remember) be `[`in square brackets`]`. Names of files will be marked in coloured normal text.

1.3. Why use SLiMSearch?

Short linear motifs (SLiMs) in proteins are functional microdomains of fundamental importance in many biological systems. SLiMs typically consist of a 3 to 10 amino acid stretch of the primary protein sequence, of which as few as two sites may be important for activity, making identification of novel SLiMs extremely difficult. In particular, it can be very difficult to distinguish a randomly occurring instance of a motif from a truly functional one. Websites, such as ELM (Puntervoll *et al.* 2003), list many known motifs and provide search tools for finding known motifs in any given sequence dataset. SLiMSearch allows a similar search to be performed using any set of sequences and motifs and incorporates all the input masking options of SLiMFinder (Edwards *et al.* 2007), which have been very useful in improving *ab initio* SLiM discovery. If the dataset is small enough, the SLiMChance algorithm of SLiMFinder can be used to determine whether the motifs found are statistically over-represented given the evolutionary relationships of the input sequences.

1.4. Getting Help

Much of the information here is also contained in the documentation of the Python modules themselves. A full list of command-line parameters can be printed to screen using the **`help`** option, with short descriptions for each one:

```
python slimsearch.py help
```

General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to SLiMSearch can be found in the distributed readme.txt and readme.html files.

If still stuck, then please e-mail me (r.edwards@southampton.ac.uk) whatever question you have. If it is the results of an error message, then please send me that and/or the log file (see Chapter 2) too.

1.4.1. Something Missing?

As much as possible, the important parts of the software are described in detail in this manual. If something is not covered, it is generally not very important and/or still under development, and can therefore be safely ignored. If, however, curiosity gets the better of you, and/or you think that something important is missing (or badly explained), please contact me.

1.5. Citing SLiMSearch

Until published, please cite the SLiMSearch Website. When using predictions from accessory software, please cite the relevant papers. If using the SLiMChance assessment of statistical over-representation, please cite the SLiMFinder publication (Edwards et al. 2007):

- Edwards RJ, Davey NE and Shields DC (2007). SLiMFinder: A Probabilistic Method for Identifying Over-Represented, Convergently Evolved, Short Linear Motifs in Proteins. ***PLoS ONE***, **2**, e967.

When using RLC conservation masking (3.1.6), please cite the RLC paper:

- Davey NE, Shields DC and Edwards RJ (2009). Masking residues using context-specific evolutionary conservation significantly improves short linear motif discovery. ***Bioinformatics* Adv. Access** Jan 9 2009.

When using alignments generated using GOPHER (Edwards 2006), please cite the SLiMDisc Webserver paper (Davey et al. 2007). Disorder predictions should cite IUPRED (Dosztanyi et al. 2005).

1.6. Availability and Local Installation

SLiMSearch is distributed as a number of open source Python modules as part of the PEAT (Protein Evolution Analysis Toolkit) package. It should therefore work on any system with Python installed without any extra setup required. If you do not have Python, you can download it free from www.python.org at http://www.python.org/download/. The modules are written in Python 2.5. The Python website has good information about how to download and install Python but if you have any problems, please get in touch and I will help if I can.

All the required files should have been provided in the download zip file. Details can be found at http://bioinformatics.ucd.ie/shields/software/peat/ and the accompanying PEAT Appendices document. The Python Modules are open source and may be changed if desired, although please give me credit for any useful bits you pillage. I cannot accept any responsibility if you make changes and the program stops working, however!

Note that the organisation of the modules and the complexity of some of the classes is due to the fact that most of them are designed to be used in a number of different tools. As a result, not all the options listed in the __doc__() (`help`) will be of relevance. If you want some help understanding the way the modules and classes are set up so you can edit them, just contact me.

1.7. SLiMSearch webserver

A webserver implementation of SLiMSearch is now available at http://bioware.ucd.ie/. This features a pared down version of the main search functions, coupled to additional visualisations. Further help on the webserver can be found on the website.

2. Fundamentals

2.1. Running SLiMSearch

2.1.1. The Basics

If you have python installed on your system, you should be able to run SLiMSearch directly from the command line in the form:

```
python slimsearch.py motifs=FILENAME seqin=FILENAME
```

To run with default settings, no other commands are needed. Otherwise, see the relevant sections of this manual.

IMPORTANT: If filenames contain spaces, they should be enclosed in double quotes: *seqin="example file"*. That said, it is recommended that files do not contain spaces as function cannot be guaranteed if they do.

2.1.2. Options

Command-line options are suggested in the following sections. General details about Command-line options can be found in the PEAT Appendices document included with this download. Details of command-line options specific to SLiMSearch can be found in the distributed readme.txt and readme.html files. These may be given after the run command, as above, or loaded from one or more *.ini files (see PEAT Appendices for details).

2.1.3. Running in Windows

If running in Windows, you can just double-click the slimsearch.py file. Commands should be placed in a file called slimsearch.ini or rje.ini, which will be read upon execution. It is recommended to use the **win32=T** option.

2.2. Input

The main input for SLiMSearch is a file of motifs and one or more protein (or DNA) sequence files in fasta or UniProt DAT formats. Sequence files are specified and masked as for SLiMFinder (see SLiMFinder manual for details) using **seqin=FILE** for single files and **batch=LIST** for multiple files.

DNA motifs/sequences

SLiMSearch is designed and optimised with protein sequences and motifs in mind. DNA can be used with the **dna=T/F** option but there are probably better DNA motif tools out there and this option is currently not well supported or documented.

2.2.1. Motif Input

The recommended motif input format is "PRESTO" format, which was used for the forerunner of SLiMSearch, PRESTO. This should have a single line per motif, with the format:

```
Name Sequence # Comments
```

Comments are optional but anything after the # will be ignored. Names *must* be unique.

Alternative allowed formats include: a fasta format file with motif/peptide names and sequences in the usual fasta format; a raw list of peptides/motifs (in this case the name and sequence will the same, so the sequence must be unique); SLiMFinder output, SLiMDisc output; TEIRESIAS output; SLiM Pickings output. Additional input formats can be added on request.

In either case, the motif should be a peptide sequence using the standard single letter amino acid codes and the following regular expression rules:

- **A** = single fixed amino acid.
- **[AB]** = ambiguity, **A** or **B**. Any number of options may be given, e.g. **[ABC]** = **A** or **B** or **C**.
- **<R:m:n>** = At least **m** of a stretch of **n** residues must match **R**, where **R** is one of the above regular expression elements (single or ambiguity).

- **<R:m:n:B>** = Exactly **m** of a stretch of **n** residues must match **R** and the rest must match **B**, where **R** and **B** are each one of the above regular expression elements (single or ambiguity). E.g. match **<F:1:2:[DE]>** will match **[DE]F**, or **F[DE]**.
- **[^A]** = not **A**.
- **X** or **.** = Wildcard positions (any amino acid). Can be followed by **{m,n}** = At least **m** and up to **n** wildcards.
- **R{n}** = **n** repetitions of **R**, where **R** is any of the above regular expression elements.
- **^** = Beginning of sequence
- **$** = End of sequence
- **(R|S)** = match **R** or **S**, which are both themselves recognisable regular expressions. Unfortunately, these motifs are not currently supported by the SLiMChance statistics and, as such, any motifs in this format with be first split into variants, e.g. **(R|S)PP** would be split into **RPP** and **SPP** and each searched separately.

E.g. (1): **[IL][^P]X{3}RG** means: "leucine or isoleucine, followed by anything but proline, followed by three residues, followed by arginine followed by glycine".

E.g. (2): **^<KR:3:5>P** means: "three of the first five amino acids must be lysine or arginine; the sixth amino acid must be proline".

Note. PRESTO formally recognised additional syntax for MS-derived sequences. These are not currently supported by SLiMSearch but can still be searched using PRESTO:

- **R{m,n}** = At least **m** and up to **n** repetitions of **R**, where **R** is any of the above regular expression elements.
- **(AB|CD)** = **AB** or **CD**. For MSMS peptides (**msms=T**), this is also **BA** or **DC**.
- **(ABC)** = **ABC** in any order (**BAC**, **CAB** etc.).

2.2.2. Motif Filtering options

As well as manually editing the input files, there are a number of options for filtering the motifs/peptides that SLiMSearch reads in. These can be found in the help documentation for the rje_slimlist.py module. **minpep=X** and **minfix=X** will set the minimum number of non-wildcard and fixed positions that the motif must contain, respectively. **minic=X** will similarly restrict motifs using an information-based score, where each fixed position has a value of 1.0, each wildcard scores 0.0 and each ambiguous position has a score in between. Some motifs (or MSMS peptides) have leading or trailing wildcard positions. By default, these will be included in any search as the information on the up or downstream region may be desirable. (Because SLiMSearch returns the sequence of the matched region, these wildcards can be used to return the sequence surrounding motif occurrences and/or to better control the region included in any hydrophobicity/disorder etc. calculations.) To remove these wildcard positions, use **trimx=F**. For DNA sequences and motifs, use **dna=T**.

2.2.3. Searching with Mismatches

SLiMSearch allows motifs and peptides to be searched with a number of mismatches allowed. This is set by the **mismatch=LIST** option, where each element of the list is in the for **X:Y**, which can be used to setup "mismatch bands". This option allows **X** mismatches when the sequence has at least **Y** non-wildcard positions. *E.g.* *mismatch=1:6,2:10* would allow one mismatch for every sequence with 6+ non-wildcard positions and two mismatches for every sequence with 10+ non-wildcard positions.

Mismatches are places in every possible non-wildcard position in the motif, including degenerate positions. Where repetitions are allowed, these are applied before any mismatches are placed. *E.g.* P{2} with one mismatch would be searched with PP, PX and XP.

NB. Searching with mismatches can dramatically increase the run-time. When searching large databases it is advisable to first perform a test run with a few sequences to calculate approximate runtimes.

2.2.4. Search Database Masking

The search database(s) for SLiMSearch are identified with the **seqin=FILE** or **batch=LIST** options, as with SLiMFinder. These sequences can be masked using the same masking options as SLiMFinder (see 3.1). If batch sequence files are used, the **maxseq=X** option can be used to limit which files are actually searched.

2.2.5. Optional Input 1: Amino acid frequency files

SLiMChance can optionally take external amino acid frequency files for its probability calculations, using the **aafreq=FILE** option. This can either be a fasta format sequence file, or a plain text file containing amino acid frequencies in two columns, with the headings "AA" and "FREQ", *e.g.*:

```
AA      FREQ
A       0.055854
C       0.020012
…
Y       0.026583
```

2.2.6. Optional Input 2: Multiple Sequence Alignments

SLiMSearch is designed to be able to use the output of GOPHER (Edwards 2006) alignments of orthologues to calculate the conservation of a given motif. Alternative sources for theses alignments can be used, as long as the format is correct (see 5.3).

2.2.7. Optional Input 3: Taxonomic subgroupings

In addition to the general conservation statistics produced for the given alignments, conservation calculations can be restricted to one or more taxonomic groups (see 5.2.10).

2.2.8. Optional Input 4: Protein Clusters

If the evolutionary filter is used (**efilter=T**), SLiMSearch uses BLAST (Altschul et al. 1990) and GABLAM (Edwards & Davey 2006) to cluster related proteins into "Unrelated Protein Clusters" (UPCs) as described in the SLiMFinder Manual. UPCs are treated as units such that multiple SLiM occurrences within a UPC are treated as a single occurrence. The UPCs generated are output into a summary file (see below). If, for whatever reason, the user wishes to define their own UPCs, this file can be replaced with one in the same format. This file should be in one of the paths indicated by the **resdir=PATH** or **buildpath=PATH** options.

This might be particularly useful in cases where the homology detection by BLAST is wrong. (Either insufficiently sensitive to weak relationships or confused by low complexity regions *etc.* By default the BLAST complexity filter is on but this can be toggled with **blastf=T/F**. BLAST sensitivity can be altered using **blaste=X** to adjust the e-value threshold. ($1e^{-4}$ by default.)) See the SLiMFinder manual for more details.

2.3. Output

SLiMFinder has multiple outputs, many of which are optional. An overview of the possible outputs is given here.

The main output of SLiMSearch is a file of SLiM occurrences, *.occ.csv, and a summary file of the occurrences for each input SLiM in *.summary.csv, including the SLiMChance calculations. In addition, each motif alignments etc. can be optionally output for each dataset. Details can be found in 4.

NB. To output this file into a directory other than the run directory, use the full path in the name (*e.g.* `resfile=SLiMSearch/slimsearch.csv`).

2.3.1. Sequence relationship output

If **efilter=T** then sequence relationships are taken into account for SLiMChance calculations. The files produced at this stage are described in Chapter 4.4.

2.3.2. Extra Dataset-specific Output

If **extras=T** then a number of additional data files are created in the directory specified by **resdir=PATH** in addition to the files described above. These outputs are described in more detail in Chapter 4.5.

2.4. Commandline Options

Table 2.1 lists the commandline options for SLiMSearch. Please see also the PEAT Appendices document for additional general commandline options and the RJE_SEQ Manual for further input data options. The documentation (help) for rje_slimcalc.py also gives more details on options for additional SLiM statistic calculations and filtering (See Chapter 5). Beginners will probably want to leave the default settings unchanged.

A few list of commandline options can also be produced by running:

```
python slimsearch.py help
```

2.5. Rerunning SLiMSearch (and Pickling)

The longest part of SLiMSearch is generally the search portion, which finds the actual motifs. In order to accelerate future analyses, SLiMSearch (a) outputs a *.*.pickle file following a search if none exists, or (b) reads in an existing *.*.pickle file in place of re-searching a database with motifs. (This uses Python's "pickle" method for saving a mid-run version of the program and all its objects.) This file could be in one of the paths indicated by the **resdir=PATH** or **buildpath=PATH** options. (If created in (a), it will always be output into the **resdir=PATH** directory.) Unless running in Windows (set **win32=T**), the pickle will be compressed with gzip.

2.5.1. Pickle naming conventions

Pickles are named using both input sequence and motif filenames, allowing multiple motif files to be searched against multiple sequence files in the same results directory. Masking options are also summarised in the pickle name. (See the Masking field of the summary output.) If motifs are given directly from the commandline, the motif file identifier will be "commandline". It is therefore not recommended to run different commandline motif searches against the same dataset in the same directory unless **pickle=T**.

2.5.2. Masked dataset pickles

Because masking in itself can be quite lengthy for large datasets, it is possible to switch on additional masking pickles that are saved for input data, independent of the main pickling, using **maskpickle=T**. By default these are named after the dataset and masking options but the masking options can be replaced with user-defined text to identify specific masking settings using **masktext=X**.

Table 2.1. SLiMSearch Commandline Options.

Option	Description	Default
	Basic Input/Output Options	
motifs=FILE	File of input motifs/peptides: - Single line per motif format = 'Name Sequence #Comments' (Comments are optional and ignored) - Alternative formats include fasta, SLiMFinder & SLiMDisc output and raw motif lists.	[None]
seqin=FILE	Sequence file to search	[None]
batch=LIST	List of files to search, wildcards allowed. (Over-ruled by **seqin=FILE**)	[*.dat,*.fas]
maxseq=X	Maximum number of sequences to process	[500]
maxupc=X	Maximum UPC size of dataset to process	[0]
maxsize=X	Maximum dataset size to process in AA (or NT)	[100,000]
walltime=X	Time in hours before program will abort search and exit	[1.0]
resfile=FILE	If **FILE** is given, will also produce a table of results in resfile	[slimsearch.csv]
resdir=PATH	Redirect individual output files to specified directory	[SLiMSearch/]
buildpath=PATH	Alternative path to look for existing intermediate files	[SLiMSearch /]
dna=T/F	Motifs and sequences are DNA	[False]
force=T/F	Force re-running of BLAST, UPC generation and search	[False]
pickup=T/F	Pick-up from aborted batch run by identifying last dataset output in resfile.	[False]
	Motif Input/Reformatting Options	
reverse=T/F	Reverse the motifs (e.g. for a test comparison data set)	[False]
wildscram=T/F	Perform a wildcard spacer scrambling (e.g. for a test comparison data set)	[False]
motifout=FILE	Name of output file for reformatted/filtered SLiMs (PRESTO format)	[None]
ftout=T/F	Make a file of UniProt features for extracted parent proteins, where possible, incoroprating SLIMs [*.features.tdt]	[False]
mismatch=LIST	List of X:Y pairs for mismatch dictionary, where X mismatches allowed for Y+ defined positions	[]
motinfo=FILE	Filename for output of motif summary table (if desired)	[None]
minpos=X	Min number of defined positions	[0]
minfix=X	Min number of fixed positions for a motif to contain	[0]
minic=X	Min information content for a motif (1 fixed position = 1.0)	[0.0]
goodmotif=LIST	List of text to match in Motif names to keep (can have wildcards)	[]
nrmotif=T/F	Whether to remove redundancy in input motifs	[0]
trimx=T/F	Trims Xs from the ends of a motif	[0]
	SearchDB Options I: Input Masking	
masking=T/F	Master control switch to turn off all masking if False	[True]
consmask=T/F	Mask residues based on relative conservation.	[False]
dismask=T/F	Whether to mask ordered regions (see rje_disorder for options)	[False]
ftmask=T/F	UniProt features to mask out	[EM,DOMAIN,TRANSMEM]
imask=T/F	UniProt features to inversely ("inclusively") mask (Seqs MUST have 1+ features)	[]
compmask=X,Y	Mask low complexity regions (same AA in X+ of Y consecutive aas)	[5,8]
casemask=X	Mask Upper or Lower case of input sequence (see 3.1.5)	[None]
motifmask=X	List (or file) of motifs to mask from input sequences	[]

Option	Description	Default
metmask=T/F	Masks the N-terminal M (can be useful if termini=T)	[True]
posmask=LIST	Masks list of position-specific aas, where list = pos1:aas,pos2:aas	[2:A]
aamask=LIST	Masks list of AAs from all sequences (reduces alphabet)	[]
	SearchDB Options II: Evolutionary Filtering	
efilter=T/F	Whether to use evolutionary filtering	[True]
blastf=T/F	Use BLAST Complexity filter when determining relationships	[True]
blaste=X	BLAST e-value threshold for determining relationships	[$1e^{-4}$]
altdis=FILE	Alternative all by all distance matrix for relationships	[None]
gablamdis=FILE	Alternative GABLAM results file (!!!Experimental feature!!!)	[None]
homcut=X	Max number of homologues to allow (to reduce large multi-domain families)	[0]
	SLiMChance Options	
slimchance=T/F	Execute SLiMFinder probability method and outputs	[True]
maskfreq=T/F	Whether to mask input before any analysis, or after frequency calculations	[True]
aafreq=FILE	Use **FILE** to replace individual sequence AAFreqs (**FILE** can be sequences or aafreq)	[None]
aadimerfreq=FILE	Use empirical dimer frequencies from **FILE** (fasta or *.aadimer.tdt) (!!!Experimental!!!)	[None]
negatives=FILE	Multiply raw probabilities by under-representation in **FILE** (!!!Experimental!!!)	[None]
background=FILE	Use observed support in background file for over-representation calculations (!!!Experimental!!!)	[None]
smearfreq=T/F	Whether to "smear" AA frequencies across UPC rather than keep separate AAFreqs	[False]
seqocc=T/F	Whether to upweight for multiple occurrences in same sequence (heuristic)	[False]
	Output Options	
runid=X	Run ID for resfile (allows multiple runs on same data)	[DATE:TIME]
logmask=T/F	Whether to log the masking of individual sequences	[True]
extras=T/F	Whether to generate additional output files (alignments etc.)	[True]
targz=T/F	Whether to tar and zip dataset result files (UNIX only)	[False]
savespace=X	Delete "unneccessary" files following run (best used with targz): - 0 = Delete no files - 1 = Delete all bar *.upc and *.pickle files - 2 = Delete all dataset-specific files including *.upc and *.pickle (not *.tar.gz)	[0]
slimcalc=LIST	List of additional statistics to calculate for occurrences, out of Cons,SA,Hyd,Fold,IUP,Chg,Comp. See the documentation (help) for rje_slimcalc.py and 5Additional Statistics and Filtering for more details on options for additional SLiM statistic calculations and filtering.	[]

2.6. Accessory Applications

Depending on the masking and filtering options selected, SLiMSearch may need to use additional applications. If alignments are being made by GOPHER, please see the documentation of GOPHER for which programs are needed and how to point to them.

2.6.1. BLAST

If **efilter=T** is used, BLAST will need to be installed and the path to the BLAST files provided using **blastpath=PATH**.

2.6.2. Disorder

Disorder is controlled by the RJE_DISORDER module. By default, disorder masking uses IUPred (Dosztanyi *et al.* 2005) (installation specified by **unipath=PATH**) with the "short" method and a cut-off of 0.2. These options may be altered (Table 2.2). If IUPred is correctly installed and the IUPred_PATH environment variable is set, no other options are needed. If not (and/or this is meaningless gibberish to you) try using the **iuchdir=T** option.

By default, each residue is treated individually and "regions" of order or disorder may be as small as 1aa long. To alter this, use the **minregion=X** option, to specify the minimum length of a region. Small regions are removed in size order, working N-terminal to C-terminal for each length. (e.g. if *minregion=3* then all 1aa regions are removed, then all 2aa regions are removed, starting with the N-terminus in each case.)

Table 2.2. RJE_DISORDER Commandline Options.

Option	Description	Default
disorder=X	Disorder method to use (iupred/foldindex/parse).	[iupred]
iucut=X	Cut-off for IUPred results.	[0.2]
iumethod=X	IUPred method to use (long/short).	[short]
iupath=PATH	The full path to the IUPred executable.	[None]
iuchdir=T/F	Whether to change to IUPred directory and run (True) or rely on IUPred_PATH env variable.	[False]
minregion=X	Min. length for an ordered/disordered region.	[0]

3. SLiMSearch Details

SLiMSearch is considerably under-documented at present. More details will be added to this section with time but please also see the SLiMFinder and PRESTO manuals for further information. If you (think you) want to use SLiMSearch and find the explanations lacking/missing, then please contact me (r.edwards@southampton.ac.uk) and I will be pleased to try to help.

3.1. Input Masking

SLiMFinder masks input by replacing regions and residues with Xs. Note that SLiMFinder masking is performed after UPC definition and therefore masking will not affect the UP relationships between sequences. If you *want* to affect UP definition, then sequences must be masked manually *before* running SLiMFinder. SLiMFinder includes several masking options (below). In addition, the rje_seq module provides additional input data filters (see RJE_SEQ Manual for details). If UniProt files are not available, the unifake utility can be useful in generating files to maximise the potential of masking (See website).

3.1.1. Disorder masking

SLiMs tend to occur in disordered regions of proteins. SLiMFinder makes use of IUPRED (Dosztanyi et al. 2005) and/or FoldIndex (Prilusky et al. 2005) to predict regions of disorder. IUPRED must be installed locally, while FoldIndex can be run over the web. Residues predicted to be "intrinsically ordered" are masked out. The IUPRED -off threshold can be altered using **`iucut=X`** (0.2 by default). To use FoldIndex instead of IUPRED, use the **`disorder=foldindex`** command. You must have an active internet connection.

Because disorder masking utilises a per-residue score, there are often single residues that are just above/below the threshold in a region that is otherwise (dis)ordered. Regions can therefore be smoothed out using the **`minregion=X`** option, which stipulates the minimum number of consecutive residues that must have the same disorder state. (Dis)ordered regions smaller than this are assimilated into the neighbouring regions, starting with the smallest (1aa regions) and working up until all regions are large enough; within each region size, the sequence is traversed from N-terminal to C-terminal.

3.1.2. UniProt Features

If the input dataset is in UniProt format then features can be masked out. Any feature types given by the **`imask=LIST`** feature will be "inclusively masked", *i.e.* any sequence not part of one of these features will be masked out. Feature types given by **`ftmask=LIST`** (EM, DOMAIN, and TRANSMEM by default) will then be masked out.

3.1.3. Low Complexity Masking

SLiMFinder uses a simple complexity filter. If any amino acid occur N+ times in a stretch of L amino acids (**`compmask=N,L`**) then the central (N-2) occurrences of that amino acid are replaced with Xs. *E.g.* PFPPIPLP would become PFXXIXLP.

3.1.4. N-terminal Methionines

If using terminal motif searching (**`termini=T`**) then there is a high risk of artefactually returned `^M*` motifs due to the high occurrence of Met at position 1. The **`maskm=T`** option masks any position 1 Ms to remove this artefact. (Of course, real `^M*` motifs may be missed as a result.)

3.1.5. Masking by Case

SLiMFinder can also mask out Upper or Lower case sequences as set by the **`usecase=T`** and **`casemask=X`** option, where **`X`** is *`Upper`* or *`Lower`*. The case of the sequences can be changed by the additional option **`case=LIST`**, where **`LIST`** is the positions to switch case, starting with first lower case (*e.g.* *`case=20,-20`* will have twenty amino acids of Upper case at each end of each sequence).

Example. To search only in the C-terminal 30aa of each sequence, use:

```
python slimfinder.py casemask=Lower usecase=T case=0,-30
```

3.1.6. Relative Conservation Masking

Sites of functional importance are likely to exhibit as much, or more, conservation than those around them. SLiMFinder incorporates a masking strategy that removes residues that violate this assumption by masking out any positions that are less conserved than flanking regions as assessed using a score based on Shannon's entropy. A conservation score is first calculated for each position, *i*, of each sequence using the optional multiple sequence alignment (MSA) input of orthologues:

$$c_i = (1 - \sum f_a \log_{20}(f_a)).g_i$$

Where f_a is the frequency of each (non-X) amino acid a in the MSA column i and g_i is the proportion of sequences in i that are not gaps. This will be 1.0 for a fixed (100% conserved) residue and tend towards 0.0 for a totally variable residue in a large alignment (i.e. all 20 aa have a frequency of 1/20). The "gap penalty" reduces the score for columns with indels and includes Xs as non-gapped residues, even though these are not included in the entropy calculation. If all the non-gap sequences in a column are Xs, the amino acid frequencies from the whole alignment are used to calculate the entropy for that position.

This is then converted into a relative conservation score, r_i, which is based on the mean conservation score across a window flanking the residue, normalised by the standard deviation:

$$r_i = \frac{c_i - c_w}{SD_w}$$

Where c_w and SD_w are the mean and standard deviation respectively of the c_i scores across a window of 30aa either side of position i. The resultant score, r_i will be positive for residues that are more conserved than their flanking residues and negative for those less conserved (mean 0.0, standard deviation 1.0). Any residues that have a different disorder state from residue i are excluded from the calculation. (Ordered regions generally show a higher level of conservation than disordered regions.)

Any residues with $r_i < 0.0$ are masked out. (If there is no MSA for a protein, all residues will have a score of 0.0 and no residues will be masked.)

3.1.7. Masking pre-defined motifs

Certain commonly recurring motifs (e.g. [KR][KR] or RSRS) can dominate results from large-scale analyses. These motifs can now be masked from the input dataset using **`motifmask=X`**, where **`X`** is a file containing motifs or simply a list of motifs (see 2.2.1). Wildcard positions in such motif occurrences will *not* be masked out. (e.g. PxxP would only mask the two prolines.)

4. SLiMSearch Output

One of the features of SLiMSearch is the wealth of information available in the various outputs. For an initial inspection of the results, the basic results files (slimsearch.csv or **resfile=FILE** and slimsearch.summary.csv) should be the first port of call and contains all the results for all datasets and their summary statistics. For further evaluation of specific results, however, the additional outputs can be extremely useful. These outputs are described in detail below and are saved in the directory specified by **resdir=PATH** (SLiMSearch/ by default).

The main results files produced by SLiMSearch are the *.csv and *.summary.csv files, which are comparable to the *.occ.csv and main results table of SLiMFinder, respectively. Matches for each motif to each sequence, along with any additional statistics that have been calculated are output into the main results table. Results for each motif in each dataset are also summarised in the *.summary.csv file.

NB. To change these from comma separated CSV files to tab-delimited TDT files, either change the file extensions to .tdt or use **delimit=tab**.

4.1. Overwriting, appending and backing up results

By default, the main results file will be overwritten and a backup (optionally) saved as *.bak. If **append=T** then the file will be appended instead, allowing multiple runs to be examined together with ease. (Whatever this setting, multiple datasets within a single batch run will be output into a single file.) All dataset-specific files will be overwritten regardless of the append setting, so multiple runs should be redirected into different output directories using **resdir=PATH**. Previous results, namely the *.upc and *.*.pickle files can still be read in with the appropriate redirection by the **buildpath=PATH** option. (SLiMSearch will first look in the results directory and then in the build path. Both are set to SLiMSearch/ by default.)

4.2. Main SLiMSearch results table

The main fields for the primary SLiMSearch output are listed in Table 4.1. Additional fields are determined by the **slimcalc=LIST** setting (see Chapter 5.1).

Table 4.1. Basic Fields for Main SLiMSearch Output.

Field	Type†	Description
Dataset	Dataset	Dataset name. Generally input filename without its file extension. This will be the first part of any dataset-specific file names in the output directory.
RunID	Run	Run identifier set by **runid=X**. The date & time is used if no ID is given. This allows the results of several runs to be compiled in a single results file and easily distinguished.
Masking	Run	Summary of masking options: 'Dis' = disorder [**dismask=T**]; 'Comp' = complexity [**compask=X,Y**]; 'FT' = UniProt features [**ftmask=LIST**]; 'Inc' = inclusive features [**imask=LIST**]; 'Freq' = Mask AA frequencies [**maskfreq=T**]; 'None' = None. ***cntd over page…***
Motif	Motif	The name of the Motif
Seq	Sequence	The short name of the sequence in which the motif was found
Start_Pos	Occ	The start position in the sequence of the motif occurrence
End_Pos	Occ	The end position in the sequence of the motif occurrence
Prot_Len	Sequence	The length of the sequence
Pattern	Motif	The motif definition
Match	Occ	The particular part of the sequence matching the motif
Variant	Occ	The particular motif variant matching the sequence at this position
MisMatch	Occ	The number of mismatches at this position
Desc	Sequence	Description of sequence hit

†Field content either pertains to the specific Motif, the Dataset searched, the Sequence in which the Motif was found, the individual motif Occurrence or the Run settings that yielded those particular results.

4.3. SLiMSearch summary table

In addition to the basic information about the dataset and motif (Table 4.2), additional statistical calculations are made based on the SLiMChance algorithm of SLiMFinder. Four statistics are calculated (*N*, *E*, *p* and *pUnd*) for each of three levels of motif occurrence (Occ, Seq and UPC):

- *N* = The observed number of occurrences of the motif in the dataset
- *E* = The expected number of occurrences of the motif in the dataset, given the motif and dataset composition. (If the background=FILE option is used, motif occurrence in the background dataset will be used to generate the *E* value instead.)
- *p* = The probability of seeing the observed number of occurrence (*N*) *or more*, given the expected number of occurrences, *E*. For UPC and Seq calculations (below) the Binomial distribution is used for this calculation, using the mean probability of a motif occurrence for each Sequence/UPC, as implemented in SLiMFinder. For the occurrence calculation, the Poisson distribution is used to calculate *p* given *E*.
- *pUnd* = The probability of seeing the observed number of occurrence (*N*) *or less*, given the expected number of occurrences, *E*. For UPC and Seq calculations (below) the Binomial distribution is used for this calculation, using the mean probability of a motif occurrence for each Sequence/UPC, as implemented in SLiMFinder. For the occurrence calculation, the Poisson distribution is used to calculate *p* given *E*.
- Occ = Calculations are made based on the total number of occurrences of the motif in the dataset, irrespective of what sequences the occurrences are in and any evolutionary relationships between them.
- Seq = Calculations are made on a sequence-by-sequence basis, such that the number of occurrences within each sequence does not matter, only whether a sequence contains *any* occurrences of the motif or not. Evolutionary relationships are not considered. (This is of most use for the *N_Seq* value.)
- UPC = Calculations are made according to the number of UPC a motif occurs in, adjusting for evolutionary relationships as performed by SLiMFinder. If all sequences are unrelated (or efilter=F), this should be the same as Seq.

Example. *E_Seq* is the expected number of sequences containing 1+ occurrences of the motif, ignoring evolutionary relationships.

Table 4.2. Basic Fields for SLiMSearch Summary Output.

Field	Type†	Description
Dataset	Dataset	Dataset name. Generally input filename without its file extension. This will be the first part of any dataset-specific file names in the output directory.
RunID	Run	Run identifier set by `runid=X`. The date & time is used if no ID is given. This allows the results of several runs to be compiled in a single results file and easily distinguished.
Masking	Run	Summary of masking options: 'Dis' = disorder [`dismask=T`]; 'Comp' = complexity [`compask=X,Y`]; 'FT' = UniProt features [`ftmask=LIST`]; 'Inc' = inclusive features [`imask=LIST`]; 'Freq' = Mask AA frequencies [`maskfreq=T`]; 'None' = None. ***cntd over page…***
RunTime	Dataset	The time taken for the dataset to run (HH:MM:SS).
SeqNum	Dataset	Number of sequences in dataset.
UPNum	Dataset	Number of UPC in dataset.
AANum	Dataset	Total number of unmasked AA in dataset.
MotNum	Dataset	Number of motifs with minimum support requirement (***i.e.*** would be output if no probability cut-off.
Pattern	Motif	Pattern of SLiM.
IC	Motif	Information content.

4.4. Sequence relationship output

If **efilter=T** then sequence relationships are defined using BLAST (Altschul et al. 1990) and GABLAM (Edwards & Davey 2006). A *.self.blast file is created during this process, containing the actual BLAST alignments. This is deleted unless **extras=T**. This file is converted into a GABLAM distance matrix, which is output into a tab-delimited distance matrix file *.dis.tdt. If **extras=T** then a PHYLIP format distance matrix is also output as *.phydis.txt, which can be used to draw trees with PHYLIP (Felsenstein 2005) (using RJE_TREE (Edwards 2006) if desired). This matrix in turn is used to define Unrelated Protein Clusters (UPCs; see 4.4.1), which are output to a relationship summary file *.upc. This file lists each cluster, the number of sequences it contains, the MST-corrected size of the UPC and the short names of the sequences in the UPC. Note that when this file exists, it will be read for a dataset instead of regenerating with BLAST (unless **force=T**). This means that it is possible to create custom UPC groupings if desired by hacking this file (see 4.4.1).

4.4.1. UPC Definitions

UPC definitions are made using an all-by-all BLAST, the results of which may be found in the *.self.blast file. UPCs are then saved in the *.upc file, which is a simple text file in the form:

```
#LIG_14-3-3_1# 4 Seq; 3 UPC; 3.897 MST
UP      N       MST     Seqs
1       2       1.897   RAF1_HUMAN__P04049 M3K5_HUMAN__Q99683
2       1       1.000   BAD_RAT__O35147
3       1       1.000   MPIP3_HUMAN__P30307
```

The first row contains the dataset name, the number of sequences, the number of UPC and the MST corrected size for the whole dataset. The closer the MST value is to 1, the more related the proteins are. A totally unrelated dataset will have an MST value equal to the number of sequences. The rest of the file is a simple table of the UPCs themselves: UP = UPC identifier; N = number of sequences in UPC; MST = corrected size of UPC; Seqs = List of sequences in that UPC.

This file can be manually edited to modify the way that SLiMFinder uses the dataset. (If present, this file will be read in by SLiMFinder rather than regenerated, unless **force=T**.)

4.4.2. Distance Matrices

In addition to this file, two distance matrices are output: a plain tab delimited file *.dis.tdt and a PHYLIP (Felsenstein 2005) format *.phydis.txt file. These files contain the pairwise GABLAM sequence identities. (For the PHYLIP file, sequence names may be replaced by the number of the sequence in the input file.)

4.5. Extra Dataset-specific Output

If **extras>0** then a number of additional data files are created in the directory specified by **resdir=PATH** in addition to the files described above:

4.5.1. Sequence files (*.motifaln.fas, *.maskaln.fas, *.mapping.fas, *.masked.fas)

Assuming any motifs are found, SLiMSearch outputs a number of sequence files to aid exploration of the results:

*.motifaln.fas = Alignment of all occurrences for each returned SLiM, with the surrounding sequence context (set using **flanksize=X** [default 30])

*.maskaln.fas = Same as above but showing masking of residues

*.mapping.fas = Fasta file containing sequences for each input sequence. These sequences are present in threes for each input sequence:

1. The significant motifs found in that sequence, aligned to (2) and (3)
2. The full-length unmasked sequence
3. The full-length masked sequence
4. In addition, if alignments are used (5.3), each protein will have its homologues aligned.

*.masked.fas = The input dataset, masked.

5. SLiM Statistics and Filtering

This part of the manual is incomplete. Please feel free to experiment with the filtering options, many of which are shared with PRESTO, SLiMPickings and SLiMSearch. (These manuals may have more information.) These options are under development and should therefore be used with an element of caution. The rje_slimcalc.py module contains the fullest list of command-line options but see also rje_slimlist.py.

5.1. Additional SLiM Calculations (slimcalc)

The additional SLiM calculations implemented by SLiMFinder are controlled with the **slimcalc=LIST** option, where **LIST** is one or more of SA, Hyd, IUP, Fold, Comp and Cons. In each case, an additional column will be added to the main *.csv output (4.2) with the relevant calculation for each SLiM occurrence. In addition, a STAT_mean column will be added to the summary output (4.3), containing the mean of the relevant stat across all occurrences of the SLiM. Percentiles can also be returned in steps defined using the **percentile=X** option, giving addition STAT_pcX columns. *E.g.* *percentile=25* will return the 0th, 25th, 50th, 75th and 100th percentile in columns *_pc0, *_pc25, *_pc50, *_pc75, *_pc100. This can be useful, for example, for identifying SLiMs for which at least 50% of occurrences meet a given criteria.

5.1.1. Surface Accessibility [SA]

This is calculated using a very crude SA estimate based on Janin & Wodak (Janin & Wodak 1978) over a 7 aa window. Each amino acid gets a SA value based on it and the 3 amino acids either side. These values are then averaged over the length of the SLiM.

5.1.2. Hydropathy [Hyd]

This is calculated using the Eisenberg scale (Eisenberg et al. 1984) over an 11 aa window, centred on each amino acid. The mean is then taken across the SLiM.

5.1.3. Disorder [IUP & Fold]

The same disorder methods used for filtering are used to calculate the mean disorder across the SLiM/window (see 3.1.1.). Each amino acid gets its own disorder score, ranging from 0 (ordered) to 1 (disordered), which is then averaged over the length of the SLiM.

5.1.4. Complexity [Comp]

The complexity measure calculated by slimcalc is very crude. It is simply the number of different amino acids observed across the length of the SLiM occurrence, divided by the maximum possible number, which is the length of the motif or twenty, whichever is smaller. E.g. a PxxPx[KR] motif occurrence with a sequence PASPPR would have a complexity of 4/6 = 0.6667.

5.1.5. SLiM Conservation [Cons]

Calculating the conservation of a SLiM occurrence is a complicated business with no clear best methodology. The relatedness of the proteins in the alignment is obviously expected to impact on any calculation, as is the general conservation of the protein as a whole. In addition, there are different ways to deal with ambiguity in motif definitions and/or amino acid substitutions that change only part of the SLiM. SLiMFinder implements a number of SLiM conservation strategies and parameters, which are covered in more detail in 5.2. In each case, however, a Cons value will be produced that ranges from 0.0 (not at all conserved) to 1.0 (completely conserved). In addition, the number of homologues present in the alignment used for the calculation (HomNum), the mean global percentage identity of these homologues (GlobID) and the mean percentage identity across the motif only (LocID) will be outputted. If the protein had no homologues, HomNum will be 0.0 but the conservation will be given as an arbitrary 1.0.

5.1.6. Extending Calculations to flanking regions

For the returned occurrence statistics, a window comprising of the motif + **winsize=X** is used. If winsize < 0 then *only* the flanks are used and not the motif itself. This does not apply to SLiM conservation scores or to the Local percentage identity returned.

5.2. SLiM Conservation Calculations

An important function of SLiMFinder is the ability to calculate conservation statistics for each match, provided alignment files are provided. (see 5.3). If alignments do not exist, GOPHER (Edwards 2006) can be used to generate them (see 5.3.1). If the identified file is not actually aligned, then RJE_SEQ will try to align the proteins using MUSCLE (Edgar 2004) or ClustalW (Higgins & Sharp 1988).

For each sequence, these alignments are used to generate the global percentage identity statistic:

- GlobID = Mean global percentage identity between query protein and homologues. This is calculated direct from the alignments, excluding matches of Xs, and is the percentage of query residues that match the aligned residue in the homologue. (Note that this is an asymmetrical measurement and the percentage of the homologue that aligns with the query may be very different if the sequences are of different lengths.)

Other conservation statistics are calculated individually for each occurrence of the motif. These are based on the homologous protein sequences available *at that site*. Any homologues with masked (X) residues that coincide to non-wildcard positions of the motif occurrence will be ignored from conservation calculations. Gaps, however, shall be treated as divergence unless the **alngap=F** option is used, in which case 100% gapped regions of homologues are also ignored (see 5.2.9). These additional statistics are:

- Cons = This is the conservation score across available homologues for that occurrence
- HomNum = Number of available homologues for that occurrence
- LocID = Mean local percentage identity between query protein and available homologues across region of match

Currently, there are four main Conservation scores implemented, which can be selected with the **conscore=X** option:

5.2.1. Absolute Conservation [abs]

For absolute conservation (**conscore=abs**), SLiMFinder first identifies the regions of the alignment that correspond to matches in the Query protein. Each aligned sequence is then taken in turn and the relevant region extracted, de-gapped, and compared to the original regular expression, *i.e.* the *degenerate* motif. The conservation score is then the proportion of these homologues in which the degenerate motif is conserved (Figure 5.1). (To calculate conservation of the specific occurrence of the motif, use the **consamb=F** option.)

5.2.2. Positional Scoring [pos]

Positional scoring (**conscore=pos**) uses a graded scoring system, where each sequence gets a score between 0 (no positions conserved) and 1 (all positions conserved). Each matching amino acid contributes a score of 1.0 (if **consinfo=F** (see 5.2.7)) and the sum over all positions is divided by the number of positions. For a degenerate site (when the default **consamb=T** option is used), the sequence must match *any* possible amino acid at that site.

The scoring matrix used for this scoring can be altered using the **posmatrix=FILE** command, where FILE contains either lists of equivalent amino acids on each line (e.g. FYW would mean that any of F, Y or W would score 1.0 vs. any other of F, Y or W), or an all-by-all matrix of amino acids and their conservation scores, e.g. this might give F-F a score of 1.0 and F-Y a score of 0.5. This allows the method to be customised according to user-determined rules. In this case, the best score between the sequence and any variant of a degenerate position is used (unless **consamb=F**).

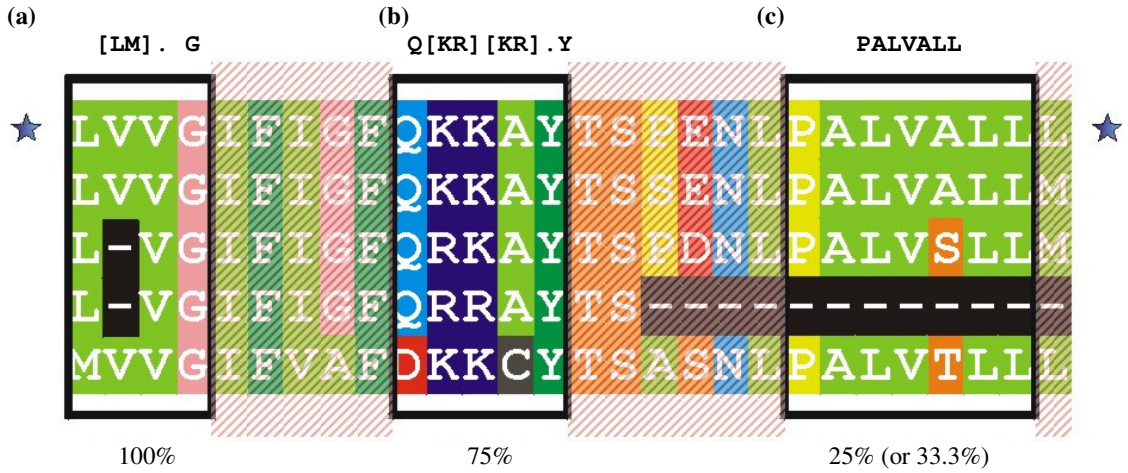

Figure 5.1. Absolute Conservation.

Three motifs are found in the query protein (marked with blue stars). This protein is ignored for conservation statistics. The black boxes represent the region of the alignment considered for each match. These matches in the homologues are then compared to the original regular expression. (a) Motif [LM]X{1,2}G is 100% conserved because, once gaps are removed, all four homologous sequences match the degenerate motif. (b) For motif Q[KR][KR]XY, one sequence does not match the degenerate motif and the query is excluded from the calculation, giving a conservation score of ¾ = 75%. (c) Only one of the homologues matches the motif. By default, all four sequences are considered, giving a conservation score of ¼ = 25%. If the alngap=F option is used, the 100% gapped sequence is ignored and the conservation is therefore 1/3 = 33.3%

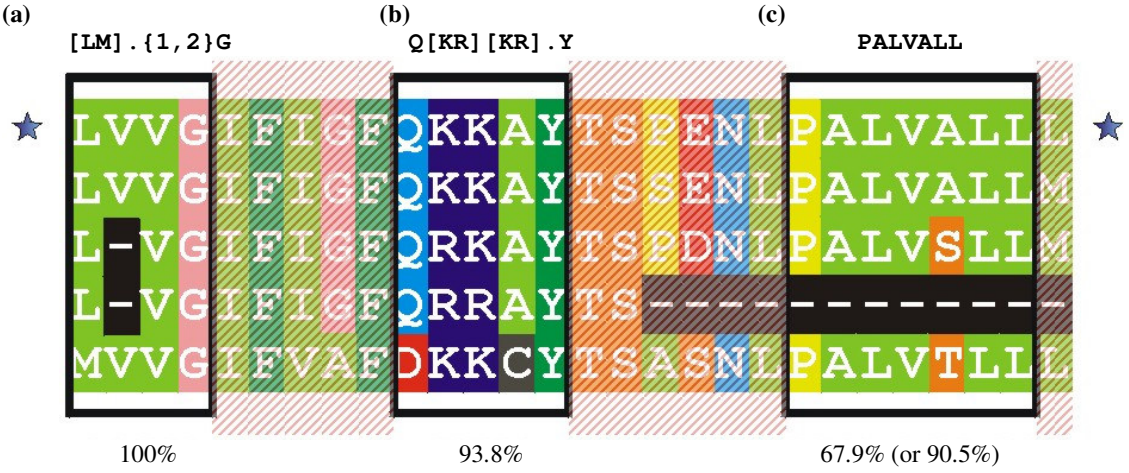

Figure 5.2. Position-specific Conservation.

The same motifs and alignments from Figure 5.1 are shown but this time position-specific conservation has been calculated. For simplicity, position-specific information content weighting has been switched off (consinfo=F) and so each position is weighted equally. (a) All positions in all orthologues still match, giving 100%. (b) Only one sequence does not match but now only one of the positions is a mismatch (D for Q) and so that sequence still gets an individual score of 75%, giving a total conservation of 93.8%. (c) Whereas the absolute conservation score penalises the two mismatches heavily, the position-specific score gives each of these sequences an individual conservation of 6/7 = 85.7%, for a total conservation score of (2x6 + 0 + 7 / 28 =) 67.9% (or 90.5% if the `alngap=F` option is used.)

5.2.3. AA Property Scoring [prop]

This (**conscore=prop**) is really just a specific example of the **posmatrix=FILE** command, where an amino acid property matrix (**aaprop=FILE**) is converted into a similarity matrix ranging from 0.0 to 1.0. By default, the property matrix of Livingstone and Barton is used (aaprop.txt) (Livingstone & Barton 1993). See the PEAT Appendices for more information on this matrix.

5.2.4. Relative Conservation Scoring [rel]

The same calculation as used for conservation masking (3.1.6) can also be used to calculate a conservation score for the motif. This is exception to the 0 < Cons < 1 rule. Relative conservation is centred around 0.0, with a standard deviation of 1.0, so all positive scores are good and all negative scores are bad.

5.2.5. Combined Scoring [all]

Under this scoring (**conscore=all**), the Cons output is the **mean** of the abs, pos and prop methods. In addition, statistics are generated for each of the individual scores:

- Pos_Cons = Positional conservation score across homologues.
- Abs_Cons = Absolute conservation score across homologues.
- Prop_Cons = Property-based conservation score across homologues.
- Rel_Cons = Relative conservation score across homologues.

The same additional options are applied to all methods, with the exception that Positional Weighting by Information Content has no effect on the absolute conservation method.

5.2.6. Motif ambiguity

By default, conservation will use the full degeneracy of the input motif. If **consamb=F** is used, the particular matching variant will be used instead. *E.g.* in Figure 5.1(b), the conservation of QKXXY would be calculated, rather than Q[KR]XXY.

5.2.7. Positional Weighting by Information Content

The **consinfo=T** option (the default) weights the contribution of different positions of the motif proportionally to their information content (IC). The IC of a position ranges from 0 for a wildcard position to 1 for a fixed position (see **Error! Reference source not found.**). For a fully fixed motif, all positions will have equal weighting. Otherwise, ambiguous positions make a smaller contribution to the score, which is normalised such that a sequence that is conserved at every position of the motif gets a score of 1.0. If **consinfo=F**, all positions contribute equally. If the "Absolute" motif conservation score is used, this weighting has no affect.

5.2.8. Homology Weighting

The **consweight=X** option controls how the conservation scores are weighted according the similarity of the homologues to the query. For each sequence s, the weighting W_s is calculated using the global percentage identity of the query versus that sequence, I_s, raised to the power of the **consweight=X** option, ω:

$W_s = I_s^{\omega} / \Sigma W_s$

When ω=0 (the default), W_s = 1 and all sequences are treated equally.

For ω=1, $W_s = I_s$, which up-weights the contribution of sequences closely related to the query. This means that the comparison of conservation scores will tend to penalise divergence in closely related sequences and will not be so heavily influenced by incorrect orthology assignment of distantly-related sequences. This weighting can be increased further with $\omega > 1$.

For ω=-1, $W_s = 1/I_s$, which up-weights the contribution of sequences distantly related to the query. This means that the comparison of conservation scores will tend to promote conservation in distantly related sequences but may be influenced by incorrect orthology assignment, which tends to be more of a problem for distantly-related sequences. This weighting can be increased further with $\omega < -1$.

5.2.9. Gap Treatment

By default, motifs that match to 100% gaps in a homologue are assumed to be due to missing or truncated sequences (as in the case of a draft genome, for example,) and do not contribute toward the relevant calculation (see Figure 5.1(c) and Figure 5.2(c)). Note that the calculation used pretends that these homologues are not present at all, and does not count them as conserved. If **alngap=T**, however, such sequences will be treated as divergence away from a motif and reduce the score. Sequences that are entirely Xs across the motif are always ignored.

5.2.10. Taxonomic subgroupings

In addition to the general conservation statistics produced for the given alignments, conservation calculations can be restricted to one or more taxonomic groups. This is achieved using the **conspec=LIST** option, where **LIST** is a list of files containing the UniProt species codes for the relevant grouping. Wildcards are allowed. Conservation analysis is then limited to these species and additional columns produced in the output (see below).

e.g. If only interested in the model organisms Human, Mouse, Rat, Chicken and Xenopus, one could use the command *conspec=model.spec_code* (or *conspec=*.spec_code*), where model.spec_code contains the species codes:

```
HUMAN
MOUSE
RAT
CHICK
XENLA
```

This would then produce additional output columns MODEL_Cons, MODEL_HomNum, MODEL_GlobID and MODEL_LocID. Where multiple files were given, each file would have its own set of output columns.

NB. The name all is reserved as a special key. Do *not* use *conspec=all.spec_code*.

5.3. Protein Alignments for SLiMFinder

SLiMFinder is designed to be able to use the output of GOPHER (Edwards 2006) for alignments of orthologues. Alternative sources for theses alignments can be used, as long as the format is correct.

Alignments should be in FASTA format with descriptions on one line followed by one or more lines containing the sequence. All sequences should be of the same length. The first word in each description should be unique. *e.g.*

```
>Seq1 And its description
SEQUENCE-ONE-GOES-HERE
>Seq2
---GAPS--ARE--ALLOWED-
>Seq3
---BUT---ALL-SEQUENCES
>Seq4
MUST-BE-EQUAL--LENGTHS
```

The file should be named AccNum.X, where AccNum is the accession number of the relevant protein in the search database, and X is given by the command **alnext=X**. Files should be found in a directory identified with the **alndir=PATH** command. The function to look for and use these alignments can be switched on using the **usealn=T** option.

5.3.1. Using GOPHER to make orthologue alignments

The program GOPHER (Edwards 2006) is provided in the download and can be called as part of the SLiMFinder search using the **usegopher=T** option. GOPHER will generate its usual files in the directory specified by the **gopherdir=PATH** option. This will generate a subdirectory named ALN, which will be set as the **alndir=PATH** parameter, if not already set. If not already set as such, the **alnext=X** option will be set to orthaln.fas.

To use GOPHER to generate alignments for SLiM conservation, you will need:

1. BLAST, MUSCLE and CLUSTALW installed on your system.
2. A sequence database containing potential orthologues. This should be identified to SLiMFinder using the **orthdb=FILE** option.

Details of how GOPHER works can be found in the GOPHER documentation.

5.4. Filtering output using SLiM Calculations

Results can be filtered at two different levels using the same basic syntax: individual motif occurrences can be rejected based on occurrence statistics, or whole motifs can be rejected based on whole-motif or combined occurrence statistics. These are controlled by the **occfilter=LIST** and **slimfilter=LIST** commands, respectively. Any usual columns of output in either the *.occ.csv (occfilter) or main results file (slimfilter) can be used, including conservation scores and percentiles *etc.* The **LIST** is in the form "`stat1>a,stat2<b,stat3=c,stat4!=d`" etc. and should either be a comma delimited list given on the commandline, or contained in a separate file (named **LIST** e.g. *statcut=my_statcut_list.txt*). If not a file name, enclose in double quotes or the <> symbols will try to pipe input/output!

The alllowed operators are:

Operator	Description
>	Filtered if the stat exceeds the cut-off
>= or =>	Filtered if the stat equals or exceeds the cut-off
<	Filtered if the stat is lower than the cut-off
<= or =<	Filtered if the stat is lower than or equal to the cut-off
= or ==	Filtered if the stat is equal to the cut-off
!= or <>	Filtered if the stat is not equal to the cut-off

All these may be applied to any stat, included text fields. Stat names should match the column headers of the output (case-insensitive). If a stat is given that is not recognised, SLiMFinder will report an error but continue processing without that stat cut-off. Note that the occurrences/SLiMs that meet the given criteria are removed (filtered).

Warning! Applying > or < to strings (*i.e.* non-numerical attributes) should be used with caution, though Python does seem to process them consistently with alphabetical sorting.

5.5. SLiM calculation/filtering options

Table 5.1 gives a summary of the main SLiM calculation and filtering options. Please check module documentation for latest developments.

Table 5.1. SLiM calculation/filtering options.

Option	Description	Default	Manual
slimcalc=LIST	List of additional attributes to calculate for occurrences - Cons,SA,Hyd,Fold,IUP,Chg,Comp	[]	5.1
winsize=X	Used to define flanking regions for calculations. If negative, will use flanks *only*	[0]	5.1.6
percentile=X	Percentile steps to return in addition to mean	[0]	5.1
usealn=T/F	Whether to search for and use alignemnts where present.	[False]	5.3
alnext=X	File extension of alignment files, AccNum.X (checked before Gopher used).	[False]	5.35.3.1
usegopher=T/F	Use GOPHER to generate missing orthologue alignments.	[False]	7.3.1
gopherdir=PATH	Path from which to call Gopher (and look for PATH/ALN/AccNum.orthaln.fas)	[./]	7.3.1
fullforce=T/F	Whether to force regeneration of alignments using GOPHER.	[False]	7.3.1
orthdb=FILE	File to use as source of orthologues for GOPHER.	[]	7.3.1
conscore=X	Type of conservation score used: - abs = absolute conservation of motif using RegExp over matched region - pos = positional conservation: each position treated independently - prop = conservation of amino acid properties - all = all three methods for comparison purposes	[pos]	5.2
conspec=LIST	List of species codes for conservation analysis. Can be name of file containing list.	[None]	2.2.7
consamb=T/F	Whether to calculate conservation allowing for degeneracy of motif (True) or of fixed variant (False)	[True]	5.2.6
consinfo=T/F	Weight positions by information content (does nothing for conscore=abs)	[True]	5.2.7
consweight=X	Weight given to global percentage identity for conservation, given more weight to closer sequences - 0 gives equal weighting to all. Negative values will upweight distant sequences.	[0]	5.2.85.2.9
alngap=T/F	Whether to count proteins in alignments that have 100% gaps over motif (True) or (False) ignore as putative sequence fragments. (NB. All X regions are ignored as sequence errors.)	[False]	7.2.8
posmatrix=FILE	Score matrix for amino acid combinations used in pos weighting. (conscore=pos builds from propmatrix)	[None]	5.2.25.2.3
aaprop=FILE	Amino Acid property matrix file.	[aaprop.txt]	7.2.3
slimfilter=LIST	List of stats to filter (remove matching) SLiMs on, consisting of X*Y: - X is an output stat (the column header), - * is an operator in the list >, >=, !=, =, >= ,< - Y is a value that X must have, assessed using *. !!! Remember to enclose in "quotes" for <> filtering !!!	[]	5.4
occfilter=LIST	Same as slimfilter but for individual occurrences.	[]	5.4

6. Appendices

6.1. Troubleshooting & FAQ

There are currently no specific Troubleshooting issues arising with SLiMSearch. Please see general items in the PEAT Appendices document and contact me if you experience any problems not covered.

6.2. References

Altschul SF, Gish W, Miller W, Myers EW & Lipman DJ (1990). "Basic local alignment search tool." ***J Mol Biol*** **215**(3): 403-10.

Davey NE, Edwards RJ & Shields DC (2007). "The slimdisc server: Short, linear motif discovery in proteins." ***Nucleic Acids Res*** **35**(Web Server issue): W455-9.

Dosztanyi Z, Csizmok V, Tompa P & Simon I (2005). "Iupred: Web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content." ***Bioinformatics.*** **21**(16): 3433-4.

Edgar RC (2004). "Muscle: A multiple sequence alignment method with reduced time and space complexity." ***BMC Bioinformatics*** **5**(1): 113.

Edwards RJ. (2006). "Presto: Peptide regular expression search tool." from http://bioinformatics.ucd.ie/shields/software/presto/.

Edwards RJ. (2006). "Gopher: Generation of orthologous proteins using high-throughput evolutionary relationships." from http://bioinformatics.ucd.ie/shields/software/gopher/.

Edwards RJ & Davey NE. (2006). "Gablam: Global alignment from blast local alignment modules." from http://bioinformatics.ucd.ie/shields/software/gablam/.

Edwards RJ, Davey NE & Shields DC (2007). "Slimfinder: A probabilistic method for identifying over-represented, convergently evolved, short linear motifs in proteins." ***PLoS ONE*** **2**(10): e967.

Eisenberg D, Schwarz E, Komaromy M & Wall R (1984). "Analysis of membrane and surface protein sequences with the hydrophobic moment plot." ***J Mol Biol*** **179**(1): 125-42.

Higgins DG & Sharp PM (1988). "Clustal: A package for performing multiple sequence alignment on a microcomputer." ***Gene*** **73**(1): 237-44.

Janin J & Wodak S (1978). "Conformation of amino acid side-chains in proteins." ***J Mol Biol.*** **125**(3): 357-86.

Livingstone CD & Barton GJ (1993). "Protein sequence alignments: A strategy for the hierarchical analysis of residue conservation." ***Comput Appl Biosci*** **9**(6): 745-56.

Prilusky J, Felder CE, Zeev-Ben-Mordehai T, Rydberg EH, Man O, Beckmann JS, Silman I & Sussman JL (2005). "Foldindex: A simple tool to predict whether a given protein sequence is intrinsically unfolded." ***Bioinformatics.*** **21**(16): 3435-8.

Puntervoll P, Linding R, Gemund C, Chabanis-Davidson S, Mattingsdal M, Cameron S, Martin DM, Ausiello G, Brannetti B, Costantini A, Ferre F, Maselli V, Via A, Cesareni G, Diella F, Superti-Furga G, Wyrwicz L, Ramu C, McGuigan C, Gudavalli R, Letunic I, Bork P, Rychlewski L, Kuster B, Helmer-Citterich M, Hunter WN, Aasland R & Gibson TJ (2003). "Elm server: A new resource for investigating short functional sites in modular eukaryotic proteins." ***Nucleic Acids Res*** **31**(13): 3625-30.