SLiMPickings: SLiMDisc results compiler and extractor

Richard J. Edwards (2007)

1: Introduction

This manual is designed to give an overview of the slimpickings.py module. Because there are many options, this manual will probably not be totally comprehensive but will hopefully cover the basics and the most useful of the more advanced stuff. If anything is missing or needs clarification, please contact me. Because it is designed primarily for internal use, this manual has a slightly different layout to my other software manuals. **Section 2: Fundamentals** will give the suggested parameter settings for common tasks as well as input and output details. **Section 3: SLiM Statistics** will give more details on how the current 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 Slim Pickings can be found in the distributed readme.txt and readme.html files

NB. 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. Furthermore, many options have been added to Slim Pickings over the past few weeks and not all of them have found their way into the manual yet. 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.

Good luck.

Rich Edwards, 2006.

1.1: Version

This manual is designed to accompany **SLiMPickings** version **3.0**.

The manual was last edited on 24 January 2007.

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 red`** for user-defined portions, such as file names etc.). Command-line examples will be given in (purple) *`italicised Courier New`*. Optional parameters will (where I remember) be `[`in square brackets`]`. Names of files will be marked in normal text by (dark yellow) **Bold Times New Roman**.

1.3: SLiMPickings Overview

This is a basic results compiler for multiple SLiMDisc motif discovery datasets. There are currently the following functional elements to the module:

1.3.1: Compilation of results from multiple datasets

This will search through the current directory and any subdirectories (unless `subdir=F`) and pull out results into a single comma-separated file (slimdisc_results.csv or `outfile=FILE`). With the basic run, the following statistics are output:

['Dataset','SeqNum','TotalAA','Rank','Score','Pattern','Occ','IC','Norm','Sim']

This file can then be imported into other applications for analysis. (E.g. rje_mysql.py can be run on the file to construct a BUILD statement for MySQL, or StatTranfer can convert the file for STATA analysis etc.)

NB: If multiple datasets (e.g. in subdirectories) have the same name, slim_pickings will become confused and may generate erroneous data later. Please ensure that all datasets are uniquely named.

1.3.2: Optional SLiM Stats

Additional optional stats based on the motifs sequences themselves to help rank and filter interesting results. These are:

- AbsChg : Number of charged positions [KRDE]
- NetChg : Net charge of motif [KR] - [DE]
- BalChg : Balance of charge = Net charge in first half motif - Net charge in second half
- AILMV : Whether all positions in the motif are A,I,L,M or V. (True/False)

1.3.3: Optional Occurrence Statistics (using PRESTO)

Calculation of additional statistics from the input sequences, using PRESTO. These are:

- Mean Surface Accessibility around the motif occurrences (including an extended window either side)
- Mean Eisenberg Hydrophobicity around the motif occurrences (including an extended window either side)
- SLiM conservation across orthologous proteins. (This calculation needs improving.)

The mean for all occurrences of a motif will be output. In addition, percentile steps can be used to assess motifs according to selected threshold criteria (in another package). This will return the threshold at a given percentile, e.g. SA_pc75=2.0 would mean that 75% of occurrences have a mean Surface Accessibility value of 2.0 or greater. (For hydrophobicity, Hyd_pc50=0.3 would be 50% of occurrences have mean Hydrophobicity of 0.3 or *less*. This is because low hydrophobicity is good for a (non-structural) functional motif.)

1.3.4: Collation and extraction of key data for specific results

These may be by any combination of protein, motif or dataset. If a list of datasets is not given, then all datasets will be considered. (Likewise proteins and motifs.) To be very specific, all three lists may be specified (`slimlist=LIST protlist=LIST datalist=LIST`).

Information is pulled out in a two-step process:

1. The slimpicks.*.index files are consulted for the appropriate list of datasets. If missing, these will be regenerated. (slimpicks.motif.index and slimpicks.protein.index both point to dataset names. slimpicks.dataset.index points these names to the full path of the results.) Only datasets returned by all appropriate lists will be analysed for data extraction.

2. The appropriate data on the motifs will be extracted into a directory as determined by `outdir=PATH`. Depending on the options selected, the following (by default all) data is returned:
    a. *.motifaln.fas = customised fasta file with motifs aligned in different sequences, ready for dotplots and manual inspection for homology not detected by BLAST.
    b. *.dat = UniProt DAT file for as many parents as possible.

These files will be saved in the directory set by `outdir=PATH`.

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 slimpickings.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 Slim Pickings can be found in the distributed readme.txt and readme.html files.

If still stuck, then please e-mail me (richard.edwards@ucd.ie) whatever question you have. If it is the results of an error message, then please send me that and/or the log file (see **Output**) too.

1.5: Availability and Local Installation

SLiMPickings can be run from the bioware webserver, available at http://bioware.ucd.ie/.

SLiMPickings is also 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 – simply copy the relevant files to your computer and run the program (**2.1: Running SLiMPickings**, below.)

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 2.4. 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.

2: Fundamentals

2.1: Running SLiMPickings

2.1.1: The Basics

If you have python installed on your system (see 1.5: Availability and Local Installation), you should be able to run XXX directly from the command line in the form:

```
python slimpickings.py
```

python /home/richard/Python_Modules/slim_pickings.py

For the most basic function of indexing and compiling all the SLiMDisc results in all subdirectories, simply enter the chosen directory and run the above command without any additional parameters. For other common tasks, see the section below, which gives the necessary additional parameters.

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 Slim Pickings 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).

2.2: Common Tasks

This section will try to cover what I perceive to be the most common tasks to be performed with slim_pickings and give suggested option settings. If this gives unexpected results, please check the Input section for possible problems with your input files and check the other relevant sections of this manual, which have more information.

2.2.1: Indexing SLiMDisc results for later extraction

Q. Do you have SLiMDisc results from one or more datasets from which you will want to extract results for specific motifs, proteins and/or datasets?

- **Yes.** Run from the parent directory for future results extraction. Use the default setting (`index=T`). This will overwrite any existing slimpicks index files.
- **No.** Switch indexing off with `index=F`.

Q. Will you be moving the whole lot (index and data files) into another directory (or machine) and still want the index files to work?

- **Yes.** Set the index files to use relative paths to data files, using `fullpath=F`.
- **No.** Use the default settings.

Q. Are you indexing a subset of all data or just want the index files in a different directory?

- **Yes.** Set the new directory for index files with `slimpath=PATH`.
- **No.** Use the default settings.

2.2.2: Compiling results from multiple datasets

Q. Do you want to compile all these results into a single file?

- **Yes.** Keep the default setting `compile=T`. Select an output filename using `outfile=FILE` (default = slimdisc_results.csv). Redirect to a different directory using `outdir=PATH`, where `PATH` is the chosen directory (absolute or relative paths should work).
- **No.** Switch off compilation using `compile=F`.

Q. Do you want to compile all the ranks returned by SLiMDisc?

- **Yes.** If this is more than 1000, increase the number of ranks returned by each dataset using `slimranks=X`, where `X` is the number of ranks to return. Otherwise, keep the default. **NB.** This will not return more ranks that SLiMDisc itself returned. If you want the top 1000 ranks, be sure to run SLiMDisc with the correct "-n X" setting.
- **No.** Use the `slimranks=X` option to limit returned data to the top X ranks of each dataset (as ranked by SLiMDisc). **NB.** If you are re-ranking data according to your own criteria, you will want to leave this setting high.

2.2.3: Additional SLiM statistics based on the pattern alone

Q. Do you want to have additional charge statistics generated for the returned patterns?

- **Yes.** Use default settings.
- **No.** Turn off unwanted stats using `abschg=F` `netchg=F` `balchg=F` and/or `ailmv=F`. (See **Output** and **SLiM Statistics** for more details of these calculations.

Q. Do you want a crude calculation of the expected occurrence of a given pattern in that dataset, providing the input dataset still exists (see Input), and a new Score based on this value?

- **Yes.** Use default settings.
- **No.** Turn this off with `expect=F`.

2.2.4: Additional SLiM statistics based on each SLiM occurrence

These calculations use parts of PRESTO and some PRESTO command-line options can be used to control them. These are also listed in this section.

Q. Are the original SLiMDisc input files still present in either the same directory as the *.rank files, or the parent directory (i.e. where SLiMDisc first found them).

- **Yes.** If the input was not *.fas or *.dat, identify the list of acceptable file extensions for slim_pickings to identify them using `inputext=LIST` (E.g. *inputext=txt,fasta*).
- **No.** These statistics cannot be generated. Do not delete/move the datasets used to generate the SLiMDisc results.

Q. Do you want to calculate Emini Surface Accessibility for each occurrence?

- **Yes.** Use default settings.
- **No.** Switch off with `slimsa=F`.

Q. Do you want to calculate Eisenberg Hydrophobicity for each occurrence?

- **Yes.** Use default settings.
- **No.** Switch off with `slimhyd=F`.

Q. Do you want SA and Hydrophobicity calculations to be based on a region of sequence either side of the actual motif (i.e. a window centred on the occurrence)?

- **Yes.** Use `winsa=X` and/or `winhyd=X` to set the number of aa either side of the occurrence to be used for calculations.

Q. Do you want to calculate Charge statistics (AbsChg, NetChg and BalChg as above) for each occurrence?

- **Yes.** Use default settings.
- **No.** Switch off with `slimchg=F`.

Q. Do you want to calculate conservation statistics for each occurrence?

- **Yes.** See a special entry below.
- **No.** Switch off with `slimcons=F`.

Q. In addition to mean statistics do you want to calculate "percentile thresholds" for each statistic? (See **SLiM Statistics** for more details.)

- **Yes.** To have 25%, 50%, 75% and 100% percentiles, use the defaults. To have a different percentile increment, use `percentile=X` (e.g. default=25).
- **No.** Set `percentile=0`.

Q. Do you want a file of each occurrence of each pattern and its individual statistics?

- **Yes.** Give a file name using `occres=FILE`. This will go in the directory determined by `outdir=PATH`.
- **No.** Use default settings.

2.2.5: Extraction of specific results

On of the best things about SLiMPickings is its ability to extract specific results for proteins, motifs and/or datasets. Note that these options are combined in the style of a Boolean "AND" operator, not "OR". I.e. if more than one type of filter is given, patterns must belong to the given list of motifs AND be found in one of the given proteins in one of the given datasets. E.g. *slimlist=RGD protlist=P03353,Q9WY7U* will return statistics for the RGD motif for any dataset in which RGD was found in P03353 or Q9WY7U.

To compile the results for these extracted data in the same way as full datasets, use the same options as above (`compile=T outdir=PATH occres=FILE`).

Q. Are the indexes for all the datasets you want to extract data for present in the current directory?

- **Yes.** Use the default settings.
- **No.** Set the directory containing the index files with `slimpath=PATH`.

Q. Do you want to extract results for specific datasets?

- **Yes.** Use `datalist=LIST` to identify datasets. These will match the names extracted during indexing and results compilation, which will correspond to the directory in which the *.dat file was found and, therefore, the * in *.dat. *E.g.* SLiMDisc will take an input dataset called mydata.fas, create a directory called mydata and results files mydata/mydata.rank and mydata/mydata.dat.rank. Slim Pickings identifies "mydata" as the dataset name.
- **No.** Use the default settings.

Q. Do you want results for specific patterns?

- **Yes.** Use `slimlist=LIST` to identify patterns that will be returned (providing they are ranked high enough (`slimranks=X`)) and obey any other datalist and/or protlist restrictions.
- **No.** Use the default settings.

Q. Do you want results for specific proteins?

- **Yes.** Use `protlist=LIST` to identify proteins that will be returned (providing they are ranked high enough (`slimranks=X`)) and obey any other datalist and/or slimlist restrictions. **NB.** These protein names must match those used in the index file (see next Q.)
- **No.** Use the default settings.

Q. Do protein names given with protlist match those in the index, which themselves match those in the *.dat.rank file?

- **Yes.** Use the default settings.
- **No.** Slim Pickings has some additional `indexre=LIST` settings, where `LIST` defines alternative matching options for mapping protlist onto the proteins analysed. These are all fudges for my own analysis but I can (probably) easily add more for other people. Current alternatives are:
    - **ipi** : will try to match a given AccNum to ipi_HUMAN__AccNum
    - **ipi_sv** : will try to match a UniProt AccNum to an IPI splice variant sequence ipi_HUMAN__AccNum-X
    - **ft** : will try to match an AccNum against SLiMDisc FullText (UniProt format) names AccNum_HUMAN
    - **ft_sv** : As ft but with splice variants AccNum-X_HUMAN

Q. Do you want a file per extracted dataset that contains alignments of all extracted motifs? (See **Output** for more details.)

- **Yes.** Use default setting of `motifaln=T`. Set the size of the flanking sequences to be returned using `flanksize=X` and the length of the dividing X region between motifs with `xdivide=X`.
- **No.** Switch off with `motifaln=F`.

Q. Do you want Slim Pickings to try and extract accession numbers of proteins containing the extracted motifs from UniProt into a UniProt file?

- **Yes.** Give a destination file using `datout=FILE`
- **No.** Use the default settings.

Q. If yes to the last question, do you want this file tabulated for easy perusal?

- **Yes.** Keep the default (`unitab=T`).
- **No.** Switch off with `unitab=F`.
- **Q.** Do you want **only** those proteins identified with `protlist=LIST` output into the motifaln and UniProt files? (By default, the filters identify motifs in datasets. These motifs must occur in one or more of the protlist proteins to be included by results are then returned for all proteins in those datasets (for those motifs).)
- **Yes.** Use `strict=T` to output reduced data. (Not recommended for most scenarios.)

- **No.** Keep the default (`strict=F`).

2.2.6: Appending or over-writing existing results

Q. Do the selected output files already exist for appending rather than over-writing?

- **Yes.** Use `append=T` to append results files rather than overwriting.
- **No.** Keep the default (`append=F`).

2.2.7: Using datasets from a fragmented directory structure

Q. Are (a) all the results in subdirectories of the current directory (depth is not important), and (b) all the *.rank files in those subdirectories desired for indexing/compilation?

- **Yes.** Use the default setting, which will pull all *.rank files in this directory and all subdirectories.
- **No.** Define the set of directories from which to pull results using the `dirlist=LIST` option, where LIST is a comma-separated list of directories, wildcards permitted, e.g. `dirlist=dir1,dir2*,dir3`. Alternatively, `LIST` is a file containing a list of directories (one per line). To avoid the program looking in subdirectories, use `subdir=F`.

2.3: Input

2.3.1: SLiMDisc Input

SLiMDisc takes as input, either a fasta file or a UniProt DAT file. (If you have another format, e.g. ClustalW aln or Phylip, you may be able to reformat it using **rje_seq.py**.) For the additional stats calculations of Slim Pickings that use PRESTO, this input file must be left in place or copied into the directory containing the *.rank files (if you want to make things 'tidier').

SLiMDisc names the output after the first part of the input name before the first '.'. E.g. my_input.fas and my_input.extra-info.dat would both be processed as **my_input**. This is what Slim Pickings recognises as the "dataset" and it looks for a file named DATASET.* for its occurrence calculations (where DATASET is the name of the dataset and * is one of the file extensions set by `inputext=LIST`.

2.3.2: TEIRESIAS Input

SLiMDisc makes a new input sequence for TEIRESIAS. If masking has been used in SLiMDisc, this file (DATASET / DATASET.fasta) will only contain the bits of sequence retained for the search. It is this file that Slim Pickings therefore tries to use for Expectation calculations. If it is missing (for example, you have run SLiMDisc with the memsaver –XT option) then Slim Picking will use the original SLiMDisc input file. This is only a problem if masking has been used.

2.3.3: SLiMDisc Output

Slim Pickings uses data from the *.rank and *.dat.rank files produced by SLiMDisc V1.2 upwards. If an earlier version of SLiMDisc was used, re-run the newest version on the data again (-TF –OF to keep your old mid-process results and make the run very quick!)

2.4: Output

2.4.1: Index Files

Index files are output into the directory specified by `slimpath=PATH`. Three files are created:

1. slimpicks.data.index contains dataset names and the corresponding path of the basic *.rank file.
2. slimpicks.prot.index contains protein names (as they are output by SLiMDisc in the *.dat.rank file) and the corresponding list of datasets that contain them. Each dataset has an entry in the first index pointing to the actual file.
3. slimpicks.slim.index has the same but for the patterns contained within the *.rank files.

2.4.2: Compiled SLiMDisc results

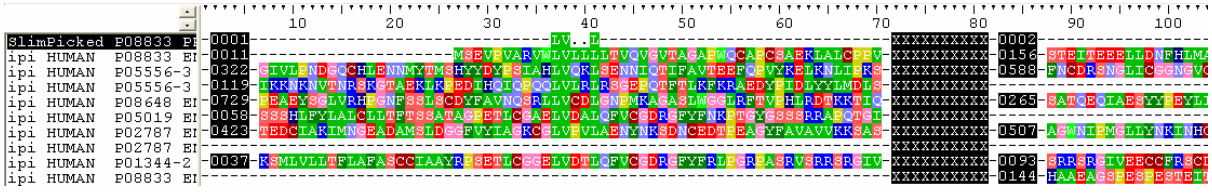
A compiled, delimited text file is produced in the directory determined by `outdir=PATH` and with the filename determined by `outfile=FILE`. The fields contained by this file are explained in the next section **3: SLiM Statistics**.

2.4.3: Motif Alignment files

Motif alignment files are named DATASET.motifaln.fas and output into the `outdir=PATH` directory. They are nominally in Fasta format and thus can be opened with viewers such as BioEdit but are not "proper" fasta (or even sequence) files and should not generally be treated as such.

These files contain each (extracted) motif from that dataset that occurs in at least one of the extracted sequences (see **2.2.5: Extraction of specific results** for details) and have the following general features (see figure, below):

- The first "sequence" entry is the motifs themselves and then below that are the occurrences of that motif, aligned around the motif itself. Note that the sequences themselves are **not** aligned, other than lining up the motif. The size of the sequence flanks is controlled by `flanksize=X`.
- The number before the motif is the position of the occurrence of that motif. (For the motif line, it is the rank of that motif in that dataset.)
- Where the same protein appears several times, it is to handle motifs that have multiple occurrences within that protein.
- Each motif is separated by a block of Xs (size set by `xdivide=X`).



At some point it will be useful to add a program for generating dotplots for these sequences to spy out shared occurrences through shared ancestry. For now, BioEdit can do this on a pairwise basis (**Sequence -> Dot plot (pairwise comparison)**) or you can select all the columns for a given motif and **Sequence -> sort -> by summed residue frequencies in selected columns** to group similar sequences together. (This works well for closely related sequences.)

2.4.4: UniProt Files

These will be described fully with documentation to come for the rje_uniprot.py module. In the meantime, they are fairly self-explanatory!

2.4.5: Log Files

The log file records information that may help subsequent interpretation of results or identify problems. Probably it's most useful content is any error messages generated. By default the log file is slim_pickings.log but this can be changed with the `log=FILE` option. Logs will be appended unless the `newlog` option is used. See the accompanying **PEAT Appendices** document for general information on log files.

2.5: Filtering, Ranking & Zscore Statistics

There are two main flavours of statistic that are generated (or used) by SLiMPickings: *Pattern Statistics* (P-Stats) generated directly from the motif pattern itself (see **3: SLiM Statistics**) and *Occurrence Statistics* (O-Stats), which are generated using data from the individual occurrences of the motifs (see **3.4: Occurrence Statistics Calculated with PRESTO**). The former are quite quick to calculate, whereas the latter are more involved and take longer to calculate. There is therefore an advantage to only calculating O-Stats on those motifs that are ultimately going to be output, *e.g.* after any filtering and re-ranking has been performed. However, sometimes it is desirable to have Z-Score statistics generated on the entire dataset before filtering occurs. This gets complicated but I will try to explain how it works here.

There are five key options to consider here:

- `rerank=X` re-ranks according to RScore (if `expect=T`) and only outputs top X new ranks (if > 0). To re-rank but still keep all motifs, set `rerank` to a very high number (>= `slimranks=X`).
- `rankstat=X` changes the stat used to re-rank data (and calculate z-scores) [RScore]
- `statfilter=`LIST list stats to filter on, with assessment conditions
- `zfilter=T/F` calculates the Z-score on the filtered dataset (True) rather than the whole dataset (False) [default = False]
- `rankfilter=T/F` re-ranks the filtered dataset (True) rather than the whole (pre-filtered) dataset (False) [default = True]. If `zfilter=T` then `rankfilter=T`.

If `rankstat` is a P-Stat (*e.g.* RScore), Z-score calculations and re-ranking of motifs occurs before any of the O-Stats are calculated unless (a) `rankfilter=T` or `zfilter=T`, and (b) one of the `statfilter` stats is an O-Stat. In this case, the re-ranking and/or z-score calculations will happen after the O-Stats are calculated.

Any P-Stat filtering will occur before any of the O-Stats are calculated unless `zfilter=T` and `rankstat` is an O-Stat. (Note that if `rankstat=Custom` and the custom stat includes an O-Stat, `rankstat` is an O-Stat.)

The ordering of these elements is summarised in the following table.

RankStat	StatFilter	ZFilter	RankFilter	Order
P-Stat	None/ P-Stat	False	False	Z-Score ⇨ Re-ranking ⇨ P-Stat Filtering ⇨ O-Stat Calculation ⇨ O-Stat filtering
O-Stat	Any	False	False	O-Stat Calculation ⇨ Z-Score ⇨ Re-ranking ⇨ P-Stat Filtering ⇨ O-Stat filtering
Any	O-Stat	False	False	O-Stat Calculation ⇨ Z-Score ⇨ Re-ranking ⇨ P-Stat Filtering ⇨ O-Stat filtering
P-Stat	Any	False	True	Z-Score ⇨ P-Stat Filtering ⇨ O-Stat Calculation ⇨ O-Stat filtering ⇨ Re-ranking
O-Stat	Any	False	True	O-Stat Calculation ⇨ Z-Score ⇨ P-Stat Filtering ⇨ O-Stat filtering ⇨ Re-ranking
P-Stat	None/ P-Stat	True	True	P-Stat Filtering ⇨ Z-Score ⇨ Re-ranking ⇨ O-Stat Calculation ⇨ O-Stat filtering
P-Stat	O-Stat	True	True	P-Stat Filtering ⇨ O-Stat Calculation ⇨ O-Stat filtering ⇨ Z-Score ⇨ Re-ranking
O-Stat	Any	True	True	P-Stat Filtering ⇨ O-Stat Calculation ⇨ O-Stat filtering ⇨ Z-Score ⇨ Re-ranking

3: SLiM Statistics

This section describes the statistics output by the Slim Pickings compilation. Note that, depending on the settings, not all statistics will be output for any given compilation.

3.1: Basic SLiMDisc statistics

These statistics are extracted directly from the SLiMDisc *.rank files.

- **Dataset.** The "name" of the dataset, matching the folder that SLiMDisc puts the results into and the base of results files (*.out *.rank etc.).
- **SeqNum.** The number of sequences in the dataset.
- **TotalAA.** The total number of amino acids in the dataset.
- **Rank.** The SLiMDisc rank of the pattern.
- **Score.** The SLiMDisc score of the pattern, used as the basis of the ranking. This is **Normalised Support x Information Content.**
- **Pattern.** The pattern returned by TEIRESIAS.
- **Occ.** The absolute support of the pattern, i.e. the total number of sequences (irrespective of evolutionary relationships) in which the pattern was found. **(NB.** This name is slightly misleading as it is **not** the number of occurrences of the pattern in the dataset but rather the number of sequences with 1+ occurrences of the pattern.)
- **IC.** Information Content of the motif, accounting to an extent for amino acid content of the input dataset. See the SLiMDisc paper for details.
- **Norm.** The normalised support for the pattern, accounting for evolutionary relationships between the sequences in which the pattern was found. See the SLiMDisc paper for details.
- **Sim.** The ratio of **Norm : Occ**, which gives a measure of the relatedness of the sequences containing the pattern. A low score indicates a high level of relatedness. A score of 1 means that no BLAST similarity was found between the sequences.

3.2: Additional pattern statistics

These are simple statistics calculated by Slim Pickings direct from the **Pattern** returned.

- **AbsChg.** The absolute charge of the pattern: **K+R+D+E**
- **NetChg.** The net charge of the pattern: **(K+R) – (D+E)**
- **BalChg.** The (crude) "charge balance" of the pattern. This is calculated by dividing the pattern into two halves, calculating the **net charge** for each half, and then subtracting the net charge of the C-terminal half from that of the N-terminal half. **NB.** With this crude measure, distance from the centre is not taken into account, so **KK…E, I.KRDS.T** and **RR.E** all have a score of 3 (2 - -1).
- **AILMV.** This is a simple True/False statistic if all the non-wildcard positions in the pattern consist of A, I, L, M or V.
- **Aromatic.** This reports the number of aromatic AAs in the motif (F,W or Y).
- **Phos.** This reports possible phosphorylation residues in the motif, i.e. S, T or Y. The type(s) of aa are reported, so possible values for this are S, T, Y, ST, SY, TY, STY, or X if there are none.

3.3: Expected Support and New RScore

In addition to the score produced by SLiMDisc, Slim Pickings calculates a new "RScore" based on the "relative support" of each pattern, derived by comparing the observed support with the expected support. This outputs three more statistics:

- **Expect. The "expected support" given the amino acid composition of the input dataset.**
- **RScore. A new score for re-ranking data: RScore = Score x Occ / Expect.**
- **Prob. A very crude estimate of the probability of observing Occ given Expect.**

These statistics are described in more detail below.

3.3.1: Expected Support

This is the "expected support" given the amino acid composition of the input dataset. Slim Pickings reads in the input dataset and calculates amino acid frequencies for each sequence independently. (NB. If present, the dataset given to TEIRESIAS is used for this calculation, so any filtering of domains etc. is also applied to the expected support calculation. If not present, the original input for SLiMDisc is used instead. If filtering has been used, this will over-inflate the expected support.)

These amino acid frequencies are then used to calculate the probability for each sequence that the pattern returned will occur in that sequence. This is calculated by first calculating the expected number of occurrences of the motif in the sequence:

- $E_{Occ} = (L - (M - 1)) \prod f(a_n)$, **where**
  - $f(a_n)$ **is the frequency of amino acid** a_n **in the sequence**
  - a_n **is the amino acid at position** n **of the motif**
  - L **is the length of the sequence**
  - M **is the length of the motif**

This is actually the equation for patterns of all fixed positions. For ambiguous patterns, $f(a_n)$ **is replaced with** $\Sigma f(a_n)$ **for all possible amino acids at position** n **of the motif.**

This expected occurrence E_{Occ} **is converted into the probability of the pattern occurring in that sequence** P_{Occ} **using a poission distribution. From wikipedia(!):**

"The probability that there are exactly k occurrences (k being a non-negative integer, $k = 0, 1, 2, ...$) is

$$f(k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!},$$

where

- e is the base of the natural logarithm ($e = 2.71828...$),
- $k!$ is the factorial of k,
- λ is a positive real number, equal to the expected number of occurrences that occur during the given interval."

The probability of 1+ occurrences is therefore 1 - the probability of zero occurrences (k**=0):**

- $P_{Occ} = 1 - e^{-Eocc}$

The expected support for the pattern across the dataset is simply the sum of the probability of 1+ occurrences in each sequence:

- **Expect = Σ P_{Occ}**

3.3.2: The new RScore

The original SLiMDisc score accounts for the length of the motif using information content and the evolutionary relationships using Normalised Support. This works well for motifs of the same length but this has a tendency to bias towards shorter motifs as the IC for a motif only increases linearly with length, whereas the background occurrence of a motif decreases exponentially. The new score attempts to account for this by using the **Expected Support** for a motif, which scales corrected with the expected background occurrence of such a motif, and calculating: **Relative Support = Observed Support / Expected Support.**

Expected Support accounts for evolutionary relationships only in the crudest sense – related proteins will have similar amino acid compositions and thus similar probabilities of having a motif. It is therefore not appropriate to compare the Expected Support with the Normalised Support of SLiMDisc, which does consider similarity. However, neither do we want to reject this information totally – accounting for these relationships is the whole point of SLiMDisc!

The compromise in the new score is to simply give the old score and Relative Support equal weighting through multiplication:

- **RScore = Score * Relative Support**

This is the same as **IC * Norm * Occ / Expect.** IC and Expect both try to account for amino acid composition and motif length in slightly different ways, upweighting unlikely patterns. Norm and Occ both try to account for slightly different measures of the support of a pattern, upweighting frequently observed patterns. The net result is hopefully a strong upweighting of **patterns occurring more often than expected and in unrelated proteins.** The new score concentrated on the "more often than expected", while the old score concentrates on "and in unrelated proteins". (The initial filtering of patterns through SLiMDisc means that all patterns returned should meet this requirement to some extent.)

3.3.3: Support Probability

In addition to the above stats, a "support probability" is calculated, again using the poisson distribution:

- **Prob = 1 - Σ $f(k;\lambda)$**, where k < **Expect** and λ = **Occ**

Note that this probability is a gross under-estimate. It does not account for "multiple testing" and the fact that only the (less probable) tail of the distribution – patterns occurring in at least 3 sequences – is being analysed. Nor does it take into account the evolutionary relationships that mean that the occurrences of the motif are not independent.

3.4: Occurrence Statistics Calculated with PRESTO

In addition to the general statistics applied to the pattern as a whole, Slim Pickings uses PRESTO to extract individual occurrences of the motif in the dataset and calculate statistics based on these occurrences:

- **SA.** This is a crude measure of Surface Accessibility based on the Emini method. (See SLiMDisc. **NB.** SLiMDisc has this as an optional filter –z [0.3 by default].) This calculation can be extended to the sequence either side of the occurrence using `winsa=X`.
- **Hyd.** This is the Eisenberg hydrophobicity for the motif in its native contexts. Again, this can be extended to the sequence either side of the occurrence using `winhyd=X`.

- **Cons.** This is a very crude measurement of conservation (where alignments are available), which is the proportion of orthologues in which the motif is 100% conserved. Note that while this measurement is appropriate for degenerate (ambiguous) motifs, it is not very suitable for the fixed-residue patterns returned by SLiMDisc and new measurements are needed.
-

4: Technical Stuff: How SLiMPickings works

This is a bit of a scary section and is really only intended for people who want to either (a) try and hack into the code to make changes, or (b) gain some insight on the correct use of options and SLiMPickings runs by understanding the underlying program architecture. In some ways, this section is really just for me!

4.1: Module Architecture

The primary SLiMPickings functions uses code from:

- slimpicking.py, contains the main SlimPicker class
- slimpicks_index.py, contains index methods for use by the SlimPicker class
- rje_motif.py and rje_motif_cons.py, contain motif classes and methods
- presto.py, contains methods for some of the occurrence stats and additional outputs
- rje_seq.py and rje_sequence.py, contain classes and methods for handling protein sequences

Some other modules are also used but these are the main ones.

4.2: SlimPicker Class Architecture

This section obvious cannot handle all the details but will give an overview.

4.2.1: Main SlimPicker Run WorkFlow

When SLiMPickings is run, it calls the run() method and sets the following approximate workflow into operation:

1. Setup of index files.
    a. The need for index files is assessed (they are needed to extract specific results) as is the pre-existence of index files in the chosen `slimpath=PATH` directory.
    b. Index files are generated (if needed) for all datasets determined by `dirlist=LIST` and `subdir=T/F`. If the `bigindex=T` option is used then a special routine for handling lots of data is called. Three index files are made: slimpicks.motif.index and slimpicks.protein.index point motifs and proteins respectively to dataset names. slimpicks.dataset.index points these names to the full path of the results (or the relative path if `fullpath=F`).
2. Setup of rank file list. If the index is used, a list of rank file names will be extracted from the dataset index, else `dirlist=LIST` and `subdir=T/F` are used to search for all rank files in the desired directories.
3. The main picks() method is called, which performs the actual processing of results.

4.2.2: Main SlimPicker Picks WorkFlow

The picks() method is called by the run() method after the list of files to process has been established and stored in list['FileList']. The picks() method peforms the following:

1. Sets up variables and output file attributes using pickSetup():
    a. This includes obj['SlimPRESTO'], a PRESTO object for storing motifs and performing calculations, and obj['SlimSeq'], a SeqList object for handling search dataset sequences and manipulations.

b. Extraction and data compilation needs are assessed from commandline options.

c. The output directory (outdir=PATH) is made if necessary and output filenames set.

d. The main compilation file is backed up if pre-existing (and not being appended).

e. The list of column headers (list['Headers']) for the compilation file is determined from options. These are divided into "PStats", which can be calculated on SLiMDisc output and Patterns alone, and "OStats", which must be calculated on specific motif Occurrences. This distinction enables calculations to be performed at the optimum time given the filtering and zscore settings, and assesses whether PRESTO calculations are needed (for OStats).

f. Custom scores are setup, checked, assigned PStat or OStat status based on the variables they use, and added to the header list.

g. Rankstat and stat filters are setup. Again, this assesses whether any OStats are needed for filtering and/or zscore calculations.

h. If GOPHER is being used for orthologue alignments, the relevant directory structures are determined and setup. (GOPHER must place alignments into an "ALN" directory, so this will be added if missing from alndir=PATH and info['GopherDir'] set to the parent directory of ALN.)

i. PRESTO and SeqList objects are setup.

j. If desired, a file for outputting data for individual occurrences is setup.

2. SLiMPickings then works through each dataset in list['FileList']:

a. The *.dat.rank file is read in and lists of motifs and proteins to process (given protlist and slimlist) made. Motifs are also added to the PRESTO object. This produces:

i. "slim_prot" = dictionary of {id:protname}

ii. "slim_motifs" = dictionary of {motif:occlist (id:pos) from dat.rank}

iii. "extract_prots" = list of proteins to extract details for

b. The *.rank file is processed to generate a dict['BasicStats'] for each motif, where the pattern is the key and the value is a dictionary of values read from the file {stat:value}. At this stage all data is still stored as strings. These are all "PStats" that require no additional calculations.

c. The original input sequence file is found and checked. If this is not fasta format and GOPHER is being used for orthologues, a fasta format file is generated for GOPHER. The TEIRESIAS input is also found and read for Expectation calculations.

d. Motif dictionaries are generated, with possible extra filtering of the extraction list if required:

i. dict['Motif PNames'] = {motif:list of protein names}

ii. dict['Motif Occ'] = {Sequence:{Motif:[positions]}}

iii. dict['SeqDict'] = {pname:Sequence object}

e. Expected Occurrence statistics are calculated for each motif, including RScore calculations, probabilities, new IC and length corrections if necessary. Also

contains two advanced probability calculations that are defunct and shall be removed shortly.

f. Post PStat filtering, ranking and Z-scores are performed. (See **2.5: Filtering, Ranking and Z-score statistics** for details.)

g. If no alignment is present and GOPHER is being used to generate alignments, these are made now.

h. PRESTO is used to generate "PRESTO Stats" for each *occurrence* of each motif, over the entire length of the motif including wildcard positions. The win*=X options determine whether windows around the motif are used. These are returned in a list "presto_seqhit", which is a list of rje_presto.PrestoSeqHit() objects.

i. PRESTO occurrence stats are combined for each motif and these "OStats" are added to the 'BasicStats' dictionary. A list of PrestoHit occurrence objects is made ("motif_hits"). Percentiles are calculated and added to the BasicStats dictionary as required.

j. Any custom OStats are made.

k. Post Ostat filtering, ranking and Z-score calculations are performed.

l. The MotifOcc dictionary is reduced to match the filtered motiflist. This is used for additional outputs.

m. Motif Occurrence statistics are output if desired.

n. Create self.dict['SlimFT'] and self.list['UniExtract'] for extraction of UniProt features with SLiMs inserted

o. Output compiled motif results for dataset

p. Output protein and motif alignments

3. Output of UniProt extractions with SLiMs inserted as features, if desired.

5: Appendices

5.1: Appendix I: Troubleshooting

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

5.2: Appendix II: References