MERCI (Motif - EmeRging and with Classes - Identification) Manual

Celine Vens[1,2], Marie-Noëlle Rosso,[2] and Etienne G.J. Danchin[2]

1 Introduction

This manual describes how to run the program MERCI. The program is available for download at `http://dtai.cs.kuleuven.be/ml/systems/merci` and is described in the following article:

> C. Vens, M.N. Rosso, and E. Danchin, *Identifying Discriminative Classification Based Motifs in Biological Sequences*, Bioinformatics 27 (9), pp. 1231-1238, 2011. (doi: 10.1093/bioinformatics/btr110)

The program comes in two files: the actual motif identification program MERCI and a program to locate occurrences of the found MERCI motifs in any sequence file. See the COPYRIGHT file for the license of this software.

2 Installing and Running the Program

The program is written in the Perl programming language, which is available from `http://www.perl.org/`. You will need to install Perl if it is not yet available on your system. Make sure to download both files MERCI.pl and MERCI_motif_locator.pl and save them in the same directory. Then go to that directory, and to run the MERCI program, type

```
perl MERCI.pl [options]
```

The MERCI_motif_locator program is called from within MERCI to locate the motifs in the positive and negative sequence set. It can also be called directly to locate the found motifs in a third sequence set by typing

```
perl MERCI_motif_locator.pl [options]
```

3 Available Options

Running the programs without any options gives an overview of the possible options. In order to find motifs, the MERCI program needs to have at least the required options set. Here, we explain all options in turn.

[1]Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Leuven, Belgium

[2]Institut National de la Recherche Agronomique, U.M.R. - I.B.S.V. INRA-UNSA-CNRS, 400 route des Chappes, BP 167, 06903 Sophia-Antipolis Cedex, France

- **-p file**: the file with the positive sequences (required). The file should be in Fasta format.
- **-n file**: the file with negative sequences (required). The file should be in Fasta format.
- **-k topK**: the number of motifs requested (values: *ALL* or any number, default: 10).
- **-fp posfreq**: the minimal frequency for the positive sequences. The frequency should be an absolute number between 0 and the total number of positive sequences (default: 1). In order to have a faster execution time, fp should preferably be set to a value higher than 1.
- **-fn negfreq**: the maximal frequency for the negative sequences. The frequency should be an absolute number between 0 and the total number of negative sequences (default: 0).
- **-o file**: output file where motifs will be stored (default: *motifs*). The output file will be overwritten, or created if it does not yet exist. The output file will contain a summarized list of options chosen, followed by the found motifs. See Section 6.
- **-l length**: the maximal motif length (default: 10000). A gap symbol counts for one in the motif length.
- **-s number**: only the first s positions of the sequences will be considered (default: 10000). Useful if, for instance, motifs specific for the N-terminal side of proteins are searched.
- **-c classification**: the classification scheme to use (values: *NONE/KOOLMAN-ROHM/BETS-RUSSELL/RASMOL* or a user defined classification file, default: *NONE*). Running the program without classification scheme (default value) assumes protein sequences, i.e., the root will be refined into the 20 amino acid residues. For use with DNA sequences, see Section 4, which also describes how to define your own classification scheme.
- **-g maxnbgaps**: maximal number of gap symbols (default: 0, no gaps). If set to a number larger than 0, then the motifs can contain variable sized gaps (denoted *gap*), of size 0 to maxgaplength.
- **-gl maxgaplength**: maximal gap length (default: 1).
- **-para parallel**: whether to run only a limited part of the search (values: 0 or a first level refinement, default: 0). See Section 5 for more information.

The program MERCI_motif_locator is called from within the program MERCI. It can also be run directly, e.g. to look for occurrences of the found motifs in a third set of sequences. These are the available options:

- **-p file**: the file to scan for occurrences of the motifs (required). The file should be in Fasta format.
- **-n file**: optional second file to scan for occurrences of the motifs (e.g. negative sequences). The file should be in Fasta format.

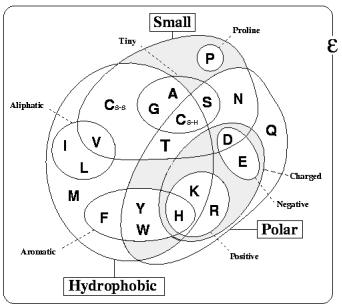

Figure 1: Classification of amino acids after Betts and Russell. The figure was taken from `http://www.russelllab.org/aas/`.

- **-i file**: input file with motifs (default: *motifs*). The file should be in the same format as the output of the MERCI program.
- **-o file**: output file where motif occurrences will be stored (default: input-file.*occurrences*). The file will be overwritten, or created if it does not yet exist. The output file contains for each motif the sequences where it occurs. It also contains the start positions of the motif. At the end, it gives a list of all sequences that are covered.
- **-s number**: only the first s positions of the sequences will be considered (default: 10000). Useful if, for instance, motifs specific for the N-terminal side of the proteins are searched.
- **-c classification**: the classification scheme to use (values: *NONE/KOOLMAN-ROHM/BETS-RUSSELL/RASMOL* or a user defined classification file, default: *NONE*). Running the program without classification scheme (default value) assumes protein sequences, i.e., the root will be refined into the 20 amino acid residues. For use with DNA sequences, see Section 4, which also describes how to define your own classification scheme.
- **-gl maxgaplength**: maximal gap length (default: 1).

4 Defining Classification Schemes

User defined classification schemes can be loaded with the option *-c filename*. We consider two examples, the classification by Betts and Russell [1], and a classification for DNA sequences.

Example 1: Classification by Betts and Russell

The classification defined by Betts and Russell [1] is shown in Fig. 1 as a Venn diagram, and the corresponding spanning tree (see article for details) is presented in Fig. 2. There are three main categories, and six smaller classes. The corresponding classification file is shown in Fig. 3.

The file contains 3 parts: a definition of the classes, a definition of the spanning tree, and the connecting DAG links. Each part starts with a title, and then lists corresponding matches.

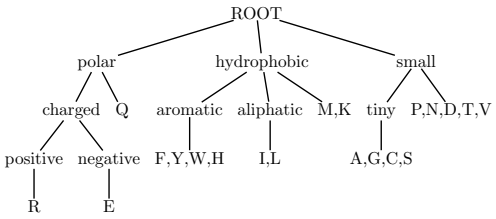

Figure 2: Spanning tree for the classification of Betts and Russell.

```
Definitions:
polar = H,K,R,D,E,Y,W,T,C,S,N,Q
charged = D,E,R,H,K
negative = D,E
positive = R,H,K
small = A,G,C,S,P,N,D,T,V
tiny = A,G,C,S
hydrophobic = H,F,W,Y,I,L,V,M,K,T,A,G,C
aromatic = H,F,W,Y
aliphatic = I,L,V

Tree structure:
root = polar,hydrophobic,small
polar = charged,Q
charged = positive,negative
positive = R
negative = E
hydrophobic = aromatic,aliphatic,M,K
aromatic = F,Y,W,H
aliphatic = I,L
small = tiny,P,N,D,T,V
tiny = A,C,G,S

DAG parents:
A = hydrophobic
C = hydrophobic,polar
D = negative
G = hydrophobic
H = positive
K = positive
N = polar
S = polar
T = hydrophobic,polar
V = aliphatic
W = polar
Y = polar
```

Figure 3: Classification file for the hierarchy by Betts and Russell.

```
Definitions:

Tree structure:
root = A,C,G,T

DAG parents:
```

Figure 4: Classification file for searching DNA motifs without classification.

- The first part is entitled *Definitions*. It contains the definitions for each class. The ordering of the classes and the corresponding amino acids is unimportant. The left side of each line contains the class name, and the right side lists the amino acids (or more generally, the symbols used in the sequences) corresponding to that class name.
- The second part is called *Tree structure*. It defines the structure of the spanning tree. Each line contains the name of a parent node, followed by a list of its child nodes. The ordering of the list is important: it is that ordering that will be used to generate candidates. If the classification scheme is a DAG, then it is important that the tree structure is such that, in depth-first traversal, a node is only visited after all its DAG parents have been visited (see article). For instance, histidine (H) can only be visited after *aromatic* and *positive* have been visited. The root element of the tree should be called *root*.
- If the classification scheme is a DAG, there is also a third part, called *DAG parents*. It lists, for each amino acid or class that has more than one parent, the parents different from its tree parent. Note that only direct parents have to be listed, no "grand parents" or other ancestors in the graph. For instance, threonine (T), belongs to the three main classes. Since it is a child of *small* in the spanning tree, only *hydrophobic* and *polar* are listed as its DAG parents. Aspartic acid (D), which is also a child of *small* in the spanning tree, is also *polar*, *charged* and *negative*. Since *negative* is more specific than *polar* and *charged*, we add it as the only DAG parent of D.

Example 2: DNA Classification

As a second example, consider DNA sequences. When searching for motifs in DNA without classes, we still have to define a classification file, otherwise the sequences are treated as proteins (by default) and MERCI will generate all 20 amino acid refinements at each step. When no classification is used, we do not have to provide *Definitions* and *DAG parents* (but do need to write the titles). The only information needed is to provide the root refinements (Fig. 4).

If we want to introduce a classification scheme into the DNA motifs, e.g. using the IUPAC nucleotide codes, then we can proceed as follows. First, we consider the directed acyclic graph (DAG) induced by the *more general than* relation between nucleotides and classes, see Fig. 5. Then, we construct a spanning tree for the DAG, such that, in depth-first (preorder) traversal, a node is visited after all its parents have been visited. A resulting spanning tree with this property is shown in Fig. 6. Finally, we construct the classification input file as shown in Fig. 7.

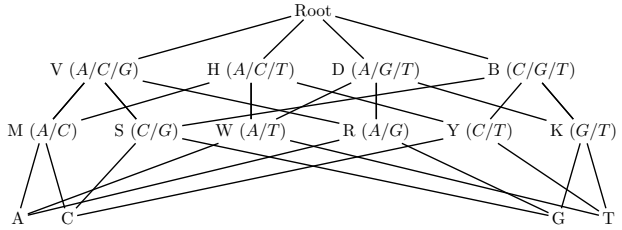

Figure 5: Directed acyclic graph for nucleotides and IUPAC codes.

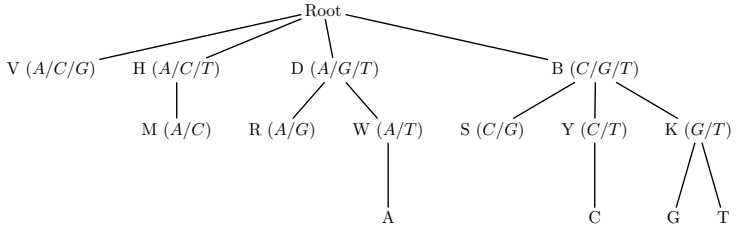

Figure 6: Spanning tree corresponding to the DAG for nucleotides and IUPAC codes.

Table 1: Part of the search space explored with -para option.

-para option	Motifs searched
-para V	motifs starting with V
-para H	motifs starting with H and M
-para D	motifs starting with D, R, W, and A
-para B	motifs starting with B, S, Y, K, C, G, and T

Table 2: Possible arguments for the *-para* option for different classification schemes.

Classification	-para arguments
KOOLMAN-ROHM	aliphatic,sulfur,aromatic,neutral,acidic,basic,P
BETTS-RUSSELL	polar,hydrophobic,small
RASMOL	neutral,acyclic,hydrophobic,large,cyclic,surface,polar,medium

5 Motif Identification in Parallel

With the *-para* option, motifs can be searched in part of the search space. As such, several independent processes can be run on a multi-processor system. The option takes one of the spanning tree's root's children as argument, and the part of the search space that is searched corresponds to all motifs starting with an element that has that argument as its ancestor in the spanning tree. For example, Table 1 shows the part of the search space explored for the previous DNA classification scheme. The possible arguments for the different classification schemes readily available in MERCI are shown in Table 2. Note that, if the top K motifs are searched (i.e., parameter $-k$ is not set to ALL), then each independent process will return K motifs, and it is left to the user to sort them in order to obtain the overall top K motifs.

```
Definitions:
R = A,G
Y = C,T
S = G,C
W = A,T
K = G,T
M = A,C
B = C,G,T
D = A,G,T
H = A,C,T
V = A,C,G

Tree structure:
root = V,H,D,B
H = M
D = R,W
B = S,Y,K
W = A
Y = C
K = G,T

DAG parents:
A = R,M
C = S,M
G = S,R
T = Y,W
M = V
R = V
W = H
S = V
Y = H
K = D
```

Figure 7: Classification file for searching DNA motifs with nucleotide codes.

6 Output

The output of MERCI consists of two files. If the option *-o* has not been set, they are called *motifs* and *motifs.occurrences*. The former file contains the motifs, together with a summary of the options that were used to obtain them. The latter file contains, for each motif, a list of sequences in the positive and negative set in which the motif occurs. For each sequence, the header is shown, the sequence id number, the start position(s) of the motif in the sequence, together with the motif variant(s) that occur(s) in the sequence. Sequences and positions are numbered starting at 1. At the end of the file, a list of all covered sequences is given.

When the MERCI program is running, motifs are printed to standard output, as they are found. With each motif, two numbers are printed: the number of occurrences in the positive and negative set, respectively. The last number can be equal to -1, indicating that the motif's frequency in the negative set was not computed. In that case, the frequency constraint was fulfilled because the motif's parent was found infrequent in the negatives. Also the current value of the F_P parameter is printed, which increases during the execution when searching for the top K motifs. After all motifs are found, timing statistics are printed.

References

[1] M.J. Betts and R.B. Russell. Amino acid properties and consequences of subsitutions. In *Bioinformatics for Geneticists*. Wiley, 2003.