

ModuleDigger-V2

ModuleDigger runs on Linux and is free to use for academic purposes only. If you use ModuleDigger in your research, please cite the following publication:

Sun H, De Bie T, Storms V, Fu Q, Dhollander T, Lemmens K, Verstuyf A, De Moor B, Marchal K. ModuleDigger: an itemset mining framework for the detection of cis-regulatory modules. *BMC Bioinformatics*.10 Suppl 1:S30.

For any non-academic purpose, please contact the corresponding author.

Please direct comments and questions related to the software to

kathleen.marchal@biw.kuleuven.be.

1. Installing the software

ModuleDigger can only be run under Linux environment. Please first download ModuleDigger, and follow the instructions described below to run ModuleDigger.

2. Software

ModuleDigger needs as input the motifs that are assigned to each gene, for instance from a motif screening. Our software consists of two steps. In a first step, all frequent cis-regulatory modules will be enumerated. In the second step, identified cis-regulatory modules will be assigned a rank.

2.1 Running motif screening tools

The user should first run a motif screening tool (we recommend Clover <http://zlab.bu.edu/clover/>) to locate the potential binding sites for each of the motifs in each gene. This should be done for both the true data set and the random data set. The true data set consist of a set of genes for which one wants to detect the cis-regulatory modules, such as a set of co-expressed genes. The random data set consists of a set of random genes. The goal of ModuleDigger is to detect cis-regulatory modules that are 2 frequently and specific for the true data set but that do not appear frequently in the random data set.

The two input files must be put in the directory `./ mineClosed/` . For each input file (both the true and random data set), each row represents a gene. Each row consists of a list of motifs that were present in the intergenic region of that gene. The motifs on each row should be space-delimited.

Example of input file:

```
motif1 motif2 motif4 motif5
motif2 motif3 motif5
motif1 motif2 motif4 motif5
motif1 motif2 motif3 motif5
motif1 motif2 motif3 motif4 motif5
motif2 motif3 motif4
... ..
```

2.2 Enumerating all frequent closed CRMs

In this step, the frequent cis-regulatory modules will be identified. The script “prepareData.sh” should be run using the input files mentioned above. The script should be run for each input file separately and will create output files in the directory ./mineClosed/.

The identified CRMs will then be statistically analyzed in the second step of the ModuleDigger algorithm (see section 2.3).

USAGE:

In a Linux platform do: **sh prepareData.sh \$1 \$2 \$3 \$4 \$5 \$6**

All arguments should be defined:

\$1 = the input file name

\$2 = the number of partitions. This parameter allows to partition the input file in smaller files such that the algorithm is faster. This should only be done in case the input file is really big (>5MB). By default, it should be set to “1”.

\$3 = the maximum percentage similarity between motif content of modules.

According to the similarity, closed modules that show this minimum percentage of similarity with other modules, will be deleted from the list. This parameter can thus be considered a filter. We choose however to set this parameter to 100 by default, such that no modules are filtered. ModuleDigger will do the filtering of the modules based on a statistical framework (see section 2.3).

\$4 = support threshold. This is the fraction of genes that should have the CRM. If there are for instance 100 genes, and at least 2 genes should have this CRM, this parameter should be set to $2/100=0.02$.

\$5 = minimum number of motifs in the module

\$6 = maximum number of motifs in the module

EXAMPLES:

We have provided two example input files: “data.txt” and “random.txt”. To identify all possible CRMs, the two following commands can be executed:

```
sh prepareData.sh data 1 100 0.01 2 3
```

Note that the file name extension “.txt” should not be mentioned when running the

algorithm. Afterwards, the following files should be present in the directory / mineClosed/ for both data.txt and random.txt:

- rows-features.txt

 This file contains information about the input file:

 first row = number of rows (genes/ sequences)

 second row = number of features (motifs)

- inputExmpl_closeds.txt

 Each row in this file contains:

 support # motifset # genelist

- inputExmpl_fmap.txt

 This file gives information about each motif. Each row consists of:

 motif's ID # support % genelist

- inputExmpl_tmap.txt

 This file provides information about rows. Each row contains:

 gene's ID : number of motifs for this gene

- map_string_inputExmpl.txt

 This file contains all ids for the motifs:

 motif : ID

These files are necessary for the second step of ModuleDigger.

2.3 Assigning a rank to each CRM

A score will be assigned to each module identified by the itemset mining approach. This score assesses how specific this CRM is for the set of genes in which it occurs, and for the cluster of input genes as a whole. Initially, the list of closed CRMs is sorted according to the scores. The CRM on top of the list is then selected as the most interesting CRM. To select the subsequent CRMs, an iterative procedure is applied. In each iteration, the score of the CRMs still in the list are updated. This score adjustment is designed to penalize CRMs that overlap with already selected CRMs, in order to make sure that selected CRMs are as nonredundant as possible with the ones that have previously been selected.

First, the top-ranked CRM is selected. Then the score of the second-ranked module is updated, and we re-rank all modules. If the updated score of this module is still ranked in the second place, this means that it does not resemble the already selected module and this CRM is considered to be interesting and non-redundant. The iteration step counter is incremented and the next iteration starts. If, however, the second-ranked CRM is not ranked as second anymore (so, it does not retain the same ranking as before updating the score), the iteration step counter is also incremented but the CRM is not selected. The CRM is then reranked according to its updated score. Subsequently the p-value of the next CRM is adjusted and checked. This iterative procedure will stop when either the maximum number of CRMs (parameter \$3) is reached or when the iteration step counter reaches its maximum (parameter \$4).

USAGE:

In a Linux environment run: **java -jar ModuleDigger.jar \$1 \$2 \$3 \$4 \$5**

\$1 = file name of the true input data

\$2 = file name of the random input data

\$3 = top \$3 number of modules that should be shown in the output

\$4 = maximum number of iteration steps (iteration step counter)

\$5 = name of the output file

EXAMPLES:

java -jar ModuleDigger.jar data random 100 10000 result

Note that the file name extension “.txt” should not be mentioned when running the algorithm.

2.4 Results

The output file of ModuleDigger will be put in .txt file in the folder ./mineClosed/. The first line indicates the number of modules that are in the output file. Each subsequent line in the output file will show a CRM, its p-value and the genes that contain the CRM: p-value # motifset % genes containing this motifset

If you run the example data of “data.txt” and “random.txt” in the folder ./mineClosed/, you will get following results.

```
covered: 82
-7.827209412010395 # motif6 motif2 motif5 % 1 3 16 24 35 36 39 57 79 81 82 84 90 91 92 104
-1.0923167943705234 # motif4 motif7 motif2 % 2 47 73 74 75 105
-0.48664309897592567 # motif2 motif9 motif5 % 1 2 3 16 24 30 31 36 39 40 51 57 78 79 80 82 87 90 91 96 116
-0.23283742528631102 # motif8 motif3 motif10 % 8 17 18 19 20 45 106 114 115
-0.10885909392392469 # motif4 motif7 motif3 % 17 18 19 66 73 74 75
-0.008119262763262828 # motif7 motif10 motif8 % 8 17 18 19 111 112 113
-0.00741547667647027 # motif4 motif6 motif2 % 1 10 35 84 86
-0.005152196139650622 # motif10 motif2 motif5 % 2 16 31 36 39 80 93 94 104
-0.0020558248053402796 # motif4 motif7 motif9 % 2 17 18 19 47 73 74 75 107
-0.0018821409468404717 # motif8 motif6 motif5 % 1 3 20 21 22 50 79 82 84 106
-0.0012488385339693212 # motif7 motif2 motif3 % 9 11 73 74 75 116
-5.658471705141519E-4 # motif2 motif1 motif5 % 1 3 24 30 31 35 36 40 41 57 78 89 92 93 94
-1.0135127923626552E-4 # motif4 motif1 motif3 % 17 18 19 56 61 66 83
-5.0421943685094705E-5 # motif10 motif6 motif3 % 8 20 36 39 83 98 106
-1.5831091694316505E-5 # motif4 motif10 motif9 % 2 17 18 19 80 83 107
-1.0372723306603138E-5 # motif7 motif8 motif1 % 17 18 19 58 59 111 112 113
-8.996017849065126E-6 # motif8 motif2 motif5 % 1 3 31 40 41 79 82 84
-5.4356248637913475E-6 # motif4 motif2 motif9 % 1 2 47 73 74 75 80
-5.357647632080565E-6 # motif4 motif3 motif9 % 17 18 19 67 68 73 74 75 83
-4.254227382711739E-6 # motif10 motif2 motif6 % 16 36 39 48 104
-2.6073221884886474E-6 # motif4 motif5 motif2 % 1 2 35 80 84
-2.5942786650543904E-6 # motif4 motif6 motif5 % 1 14 15 35 84
... ..
```