Motyl

Scientific programming Project (Python) by Flavia Leotta

Disclaimer

This is a project created for purely didactic purposes, as an assignment for the Scientific Programming course held by Professor Piro Rosario Michael. The author is Flavia Leotta, a student of MSc Bioinformatics for Computational Genomics fo the University of Milan in collaboration with Politecnico of Milan. Nevertheless, the main code for Motyl and all associated files are available for personal use with no restriction: citing the author is not necessary, but deeply appreciated.

For any questions regarding Motyl, feel free to contact me through the following e-mail: flavia.leotta@hotmail.com (mailto:flavia.leotta@hotmail.com).

Introduction

Motyl is a RESTful Web Server created with the intention of handling information about TF binding motifs taken from the JASPAR database (https://jaspar.elixir.no/). The service comes with a set of available TF (transcription factors) binding motifs but the user can freely modify the local database by using the functions provided by the service.
TFs are proteins that can bind specific DNA sequences and regulate gene expression (Thanasis Mitsis *et al.*, 2020) (https://doi.org/10.3892/wasj.2020.32) and the JASPAR database makes its mission to store these DNA sequences as position frequency matrices (PFMs). PFMs summarize occurrences of each nucleotide at each position in a set of observed TF-DNA interactions: that means that there’s not a single possible sequence for a TF binding motif, but through the PFM it is possible to predict the most probable one (consensus sequence).

The basic functionality of the Web Service are:

- Obtain data for all stored motifs;
- Obtain data for a single stored motif;
- Add a new motif to the database;
- Update an existing motif in the database;
- Delete an existing motif from the database;
- Submit a given short DNA sequence and get a match score for each motif that has the same length.

Motyl is provided in two version: a version that can be run through a command line (Motyl.py) and a web version of it (Motyl_web.py) that provides the user an interactive interface through any internet browser. In this documentation, all the functions, both the ones that are accessible to the user and the ones that shouldn’t be directly called, are shown and explained.

Directory structure

Motyl comes in a zipped file that shouldn't be further modified aside from being unzipped. Each file is essential for the correct function of the program and, although the main code is stored in Motyl.py/Motyl_web.py, all the other supporting files should be stored in the same directory.

```
Motyl/
├── database.tsv              # Database file containing tab-separated values
├── database_management.py    # Python script for database operations
├── documentation/            # Existing documentation: readme files, images used and required libraries
│   ├── Motyl_sitemap.pdf
│   ├── README.html
│   ├── README.pdf
│   └── requirements.txt
├── MotifClass.py             # Defines a class for motif-related functionality
├── Motyl.py                  # Main script for the package
├── Motyl_web.py              # Web-based interface functionality
├── saved_svg/                # Contains saved .svg files for the nucleotide frequencies (e.g.,
MA0001.1_SVG.svg)
├── static/                   # Static assets like images for the web version (e.g., logo.png)
├── templates/                # HTML templates for the web version
│   ├── database.html
│   ├── edit_motif_page.html
│   ├── matching_scores.html
│   ├── provide_sequence.html
│   ├── return_from_internet.html
│   ├── search_page.html
│   ├── single_motif_info.html
│   └── welcome_page.html
└── _pycache_/                # Compiled Python files for caching
```

Supporting files

These files support the main code for Motyl: the functions that are going to be described shouldn't be called directly by the user.

1. MotifClass.py

This file defines the class used to store the information about transcription factors (TFs) binding motifs, following the structure found in the JASPAR database.

Core Motif class

An element of class Motif is defined by 3 essential elements:

- JASPARID (*string*). The string must be of length 8, starting with 'MA' followed by 4 numbers, a dot '.' and one last number. *Ex: MA0001.1*;
- TFname (*string*). This can be of variable length;
- PFM (*Pandas dataframe*). It must be composed of 4 columns and a variable number of rows, and only contain numbers and no *Null* elements. Each column has to start with the letter associated to one of the possible nucleotides, in the correct order: 'A', 'C', 'G' and 'T'.

The attributes can be accessed using:

- `self.ID` : returns the JASPAR ID (string), a unique identifies for each motif;

- `self.PFM` : returns the Pandas dataframe containing the information about the PFM (Pandas dataframe);
- `self.name` : returns the 'TFname' (string). As far as possible, the name is based on the standardized Entrez gene symbols;
- `self.dict_info` : returns a summary of all the attributes in a more accessible way (dictionary);
- `self.TFclass` : returns the structural class of the transcription factor, based on the TFClass system (string);
- `self.family` : returns the structural sub-class of the transcription factor, based on the TFClass system (string);
- `self.collection` : returns the collection the profile belongs: CORE or UNVALIDATED (string);
- `self.taxon` : returns the taxon that the species in which this motif is found belong to. Can be one of 6 groups: vertebrate, plants, fungi, insects, urochordata and nematodes (string);
- `self.species` : returns the latin name of the specie(s) in which this motif can be found (list of strings);
- `self.dataType` : returns the methodology used for matrix construction, e.g., ChIP-seq, PBM, SELEX (string);
- `self.validation` : returns a link to PubMed indicating the orthogonal evidence of the TF binfing profile (string);
- `self.uniprotID` : returns a link to the corresponding UniProt record (integer);
- `self.source` : returns a reference to the data, which was used to build a profile for the motif (string);
- `self.comment` : returns a curator comment (if it was added in the JASPAR database) (string);
- `self.sequence` : returns the consensus sequence of the motif, computed using the PFM, as a string containing the most frequent nucleotide at each position (string).

There are some core functions associated to the Motif class:

- `__str__()` : a function that simply overwrites the `print()' function and allows to display the information in a user-friendly manner. For example:

```
JASPAR ID: MA0001.1
Name: AGL3
PFM (Position Frequency Matrix):
A : [      0      3     79     40     66     48     65     11     65      0]
C : [     94     75      4      3      1      2      5      2      3      3]
G : [      1      0      3      4      1      0      5      3     28     88]
T : [      2     19     11     50     29     47     22     81      1      6]
TFclass: MADS box factors
Family: MIKC
Collection: CORE
Taxon: Plants
Species: 'Arabidopsis thaliana'
Datatype: SELEX
Validation: 7632923
Uniprotid: -
Source: -
Comment: -
Sequence: CCATAAATAG
```

- `set_PFM(PFM)` : a function that sets a given Pandas dataframe as the PFM only if is of the format expected from a PFM;
- `validmatrix(nt)` : a nested function inside `set_PFM(PFM)` that checks if all the columns in the matrix are of the same length (have the same number of rows) and if any of the frequencies' values in the nucleotides (*nt*) columns is *Null*;
- `set_ID(JASPARID)` : similarly, a function that sets a given JASPAR ID as the motif ID only if is of the format expected ('MAXXXX.X'). It calls the function `is_valid_ID(ID)` , seen later;
- `set_sequence()` : be PFM a Pandas dataframe *L x 4* where *L* is the length of the motif and each column represents the absolute frequencies of nucleotides A, C, G and T at each position. The consensus

sequence *S* defined by this function has:

$$S[i] = \arg \max_{x \in \{A,C,G,T\}} PFM[i,x]$$

- `__setattr__(name, value)` : a function that checks if the attribute *name* has a valid *value* and, if the check passed, sets that attribute and adds it to *dict_info*.

Some functions are specific for the species attribute management:

- `add_species(species)` ;
- `remove_species(species)` .
  Both take either a string or a list of strings as argument (*species*) and either adds them or deletes them from the `self.species` attribute.

Utilities for creating motifs

In the same file there are also another two functions not specific of the class Motif but that are useful to manage this kind of elements:

- `create_motif_from_info(dictionary)` : creates an element of Motif class using a dictionary of attributes;
- `is_valid_ID(ID)` : an important function (also called inside the Motif class) that checks if the JASPAR ID is a string in the correct format.

2. database.tsv

A tab-separated file containing the motifs stored in the local memory. It is essential to never manually modify it as there are several functions implemented in the next supporting file to operate with it. It is also important to never leave it open as this creates conflicts with all the functions that manipulate the database.
The first row contains the name of the attributes specific of the class Motif (as stated in the previous section) and each subsequential row stores the information for an individual and unique JASPAR TF Motif. Each column (aside from the three essential elements ID, name and PFM) are editable through specific functions.

3. database_management.py

Contains utility functions for managing and interacting with the motif database.

Core database operations:

- `get_from_internet(ID)` : fetches a motif from the JASPAR database (https://jaspar.elixir.no/) using its ID, and creates the corresponding element of class Motif;
- `get_from_database(ID)` : similarly, retrieves a motif from the local database (database.tsv) and returns an element of class Motif;
- `add_to_database(motif)` : adds an element of class Motif to the database, by converting its information in a tab-separated string;
- `delete_from_database(ID)` : deletes a motif from the database when its ID is given as argument;
- `display_database()` : displays all the IDs of the motifs present in the local database in a tabular format.

SVG-related utilities:

- `get_svg(ID)` : retrieves the .svg file available on the JASPAR database for each motif. This .svg file contains a graphical view of the nucleotides' frequencies at each position of the file. Each letter's dimension is proportionate to its absolute frequency available in the PFM. See JASPAR documentation for more information about it;
- `get_svg_web(ID)` : retriebes the same information as the previous function, but it's the function used by the web version of Motyl.

Matching scores:

- `get_matching_scores_simple(sequence1, sequence2)` : calculates the matching scores between two sequences, with *sequence2* used as reference. The two sequences need to have the same length *L*, and the score is then computed as:

$$\text{Score} = \sum_{i=1}^{L} \delta(\text{sequence1}[i], \text{sequence2}[i])$$

where $\delta(a,b)$ is defined as:

$$\delta(a,b) = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{otherwise.} \end{cases}$$

- `get_matching_scores_sophisticated(sequence, TF)` : performs an advanced scoring between a sequence and a motif (of which we know the consensus sequence and the PFM) based on the probability of obtaining the two sequences, computed using the absolute frequencies in the PFM.
The formula is defined as:

$$\text{Score} = \prod_{i=1}^{L} P(\text{sequence}[i] \mid i)$$

where:

- L is the length of the sequence.
- $P(\text{sequence}[i] \mid i)$ is defined as:

$$P(\text{sequence}[i] \mid i) = \frac{\text{PFM}[\text{sequence}[i]][i]}{\sum_{x \in \{A,C,G,T\}} \text{PFM}[x][i]}$$

Interpretation

1. The probability of each nucleotide $\text{sequence}[i]$ being at each position i is computed using the PFM;
2. Then the frequency of that nucleotide is normalised using the sum of all frequencies at that position i;
3. Each probability is multiplied to obtain the final probability of obtaining the given sequence knowing the PFM of the motif. An higher value would mean that the given sequence has an higher probability of being one of the possible sequences found for the motif we're comparing the sequence to.

- `get_scores_from_database_simple(provided_sequence)` : this function repeats the `get_matching_scores_simple(sequence1, sequence2)` function for each entry of the database that has the same sequence length *L* as the *provided_sequence*. It returns a dictionary sorted by score (from the biggest to the smallest) with JASPAR IDs as keys and scores as values;
- `get_scores_from_database_sophisticated(provided_sequence)` : this function repeats the `get_matching_scores_sophisticated(sequence, TF)` function for each entry of the database that has the same sequence length *L* as the *provided_sequence*. It returns a dictionary sorted by score (from the biggest to the smallest) with JASPAR IDs as keys and scores as values.

4. requirements.txt

A file that contains a list of the required libraries to run Motyl. Before launching Motyl, make sure to install the required libraries by running the following command in the main directory:

```
pip install -r requirements.txt
```

5. saved_svg folder

The folder in which the .svg files are stored when downloaded.

6. templates folder

The folder containing all the .html files for the web version of Motyl. They allow the user to interact with the data in a more direct way and it is advisable to not edit them as they can directly modify the local database storing the Motifs' information.

7. static folder

The folder containing the Motyl logo, necessary for the visualization in the web interface of Motyl.

Main files

As stated before, Motyl can be used in two ways: through the command line and through a web interface. Here are described the way to use both interfaces.

1. Motyl.py - command line interface

To run Motyl on their computer, first the user needs to define the port they want to use: the default one is 5000 but the user can use any number greater than 0 and lower than 65535 and directly modify it in the first lines of the Motyl.py file. After setting the port (or accepting the default 5000), the user can launch the program from a command line interface: for example, as a Windows user, I either use Visual studio code or launch the `python Motyl.py` command on the Terminal.
When launching Motyl.py from the first command line, the user should expect to see a message similar to this:

```
* Serving Flask app 'Motyl'
 * Debug mode: on
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

This means that the server started and the URL (in this case http://127.0.0.1:5000 (http://127.0.0.1:5000)) is the URL where it operates. This is going to be the URL used for the next examples, but the user needs to use the one displayed by their own interface.
In a second command line interface (while still keeping the server running in background) the user can input commands and interact with the application. Here are the possible operations, with examples:

- **Display the database.** The database is available at route `/motifs` . Example to display it (in ascending ID order):

```
curl http://localhost:5000/motifs
```

- **Display a specific motif.** Each motif can be accessed through it JASPAR ID at route `/motifs/<JASPARID>` . This same command is able to display motifs present in the local database but, if a match is not found there, it can also look for the information from the JASPAR database and display what it has found. It can also display information for more than one motif: if the user wishes to, they can provide several IDs, separated by commas. Example to display a single motif (MA0001.1) and for displaying two motifs (MA0001.1 and MA0002.1):

```
curl http://localhost:5000/motifs/MA0001.1
curl http://localhost:5000/motifs/MA0001.1,MA0002.1
```

- **Save in the database a motif found on the internet.** The user can save a motif from the JASPAR database using the route `/save_in_database`. This command requires some additional information, but as before it can manage more than one ID. It also checks if the motif is a valid one and if it is already present in the database and, in both cases, doesn't save it and displays a message. Example that includes several IDs and also an invalid one:

```
curl -H "Content-type: application/json" -X POST -d "{\"id\":\"MA0992.1,MA0032.1,MA9999.1,MAMM
A.MIA\"}" http://127.0.0.1:5000/save_in_database
```

- **Save a .svg file of the nucleotides' frequencies.** The user can download the .svg files from the JASPAR database from the route `/motifs/<JASPARID>/svg` and, as before, more than one ID can be privded. Example:

```
curl http://localhost:5000/motifs/MA0001.1/svg
```

- **Delete a motif from the local database.** Available at route `/motifs/<JASPARID>/delete`, the user can use this function to delete more than one motif (IDs must be separated by commas) from the local database. If a motif not present in the database or an invalid ID is given, it returns a message and no action is performed. Example:

```
curl -X DELETE http://127.0.0.1:5000/motifs/MA0022.1/delete
```

- **Get matching scores for a provided sequence.** Using route `/sequence/<SEQUENCE>`, the user can obtain matching scores with motifs of the same length of a provided sequence (SEQUENCE). Example:

```
curl http://127.0.0.1:5000/sequence/CAATTAATGC
```

The user should expect this kind of result:

```
MA0990.1 has 10/10 matching bases,
for a probability of 1.527355e-02 of being your binding site
(100.0% of consensus sequence 'CAATTAATGC' probability)

MA0001.1 has 6/10 matching bases,
for a probability of 9.299204e-06 of being your binding site
(0.0258105531% of consensus sequence 'CCATAAATAG' probability)
```

The 'percentage of consensus sequence' is calculated as:

$$percentage = \frac{\text{probability of obtaining the provided sequence}}{\text{probability of obtaining the consensus sequence}} * 100$$

and the probabilities of obtaining the sequences are computed using the `get_matching_scores_sophisticated()` function from database_management.py.

- **Edit motif information (species excluded).** On route `/<JASPARID>/edit` the user can edit a motif information. This function only allows one field of one motif to be edited at time: after 'PUT -d' the user can type a dictionary with the attribute name as the key and the attribute's value as the dictionary value. The only valid attributes' names are: 'TFclass','family','collection','taxon','dataType','validation','uniprotID','source','comment' (capital letters needed). Example:

```
curl -H "Content-type: application/json" -X PUT -d '{"source":"25215497"}' http://localhost:50
00/MA0990.1/edit
```

- **Edit motif's species information.** This specific function, available at route `/<JASPARID>/edit/species` , only edits the species attributes, but it allows for different action on it. Example:

```
curl -H "Content-type: application/json" -X PUT -d "{\"delete\":\"Homo Sapiens,Mus Musculus
\"}" http://localhost:5000/MA0990.1/edit/species
```

The dictionary that the user can type to provide the data, can take as keys:

- 'add' : adds the specie(s) to the list;
- 'delete' : deletes the indicated specie(s);
- 'substitute' : completely substitutes the list of species.

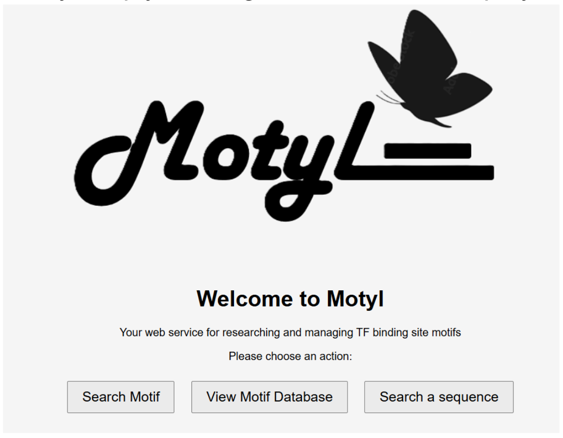
They value of the key can be either one or more species. To indicate a list of species, make sure to respect capital letters and divide them by commas: since spaces will be retained, it's recommended to not type a space after the comma, so the species will not be saved as " Mus Musculus".

2. Motyl_web.py - web interface

The web interface allows the user to do everything that can be done through the base version of Motyl but through a more user-friendly interface. It requires to be opened in a browser. The only step needed to do so is the same as the first one in the previous section: first the user needs to define the port they want to use, either the default one (5000) or any number greater than 0 and lower than 65535 that can be explicitly changed in the first lines of the Motyl.py file. After setting the port (or accepting the default 5000), the user can launch the program from a command line interface (Visual studio code or the Terminal). This allows the server to start and the URL for it will be displayed: the user can simply press CTRL and left click on the URL for it to be opened on the browser, allowing them to directly interact with Motyl's Welcome page.
For a better navigation of the site pages, a useful sitemap is given in Motyl_sitemap.pdf file in the same directory as this documentation: please refer to it for a graphical view of the routes, buttons and pages that the user can interact with.

2.1 Welcome page

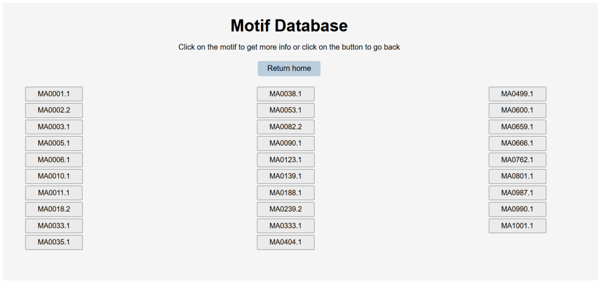
The welcome page is the first page that is shown when clicking on the URL (in the previous examples http://127.0.0.1:5000 (http://127.0.0.1:5000)): it allows the users to navigate through the main three pages of the site by simply clicking on the buttons displayed.



- 'Search motif' redirects to the page that allows the user to provide a JASPAR ID to look for;
- 'View motif database' redirects to the database page;
- 'Search sequence' redirects to the page that allows the user to provide a nucleotides sequence.

2.2 Database page

The database page simply displays the JASPAR IDs of all the motifs present in the local database: by simply clicking on a motif button, the site redirects the user to the individual motif page.

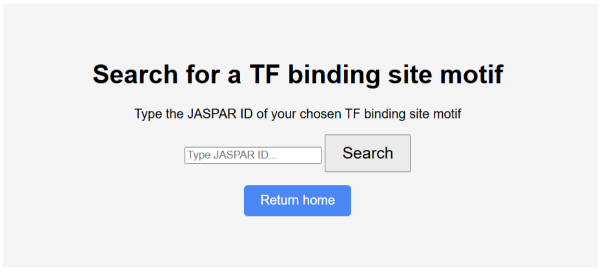


Motif Database

Click on the motif to get more info or click on the button to go back

2.3 Search motif page

This page allows the user to provide a JASPAR ID and looks for it in the local database. If a match is found, it redirects to the motif page while, if not, it looks for the motif on the JASPAR database and redirects to a slightly different version of the motif page.



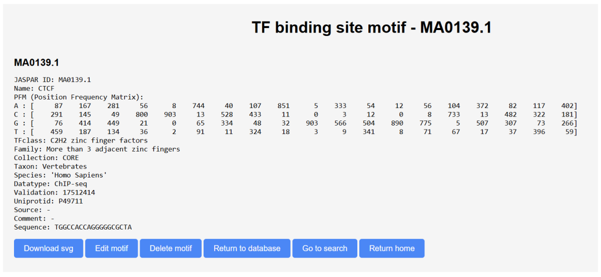
Search for a TF binding site motif

Type the JASPAR ID of your chosen TF binding site motif

2.4 Motif from database page

In this page the user can find all the information about a certain motif, as long as all the possible actions that can be performed on it:

- Download the .svg file of the nucleotides frequencies;
- Edit motif information;
- Delete motif from the database.



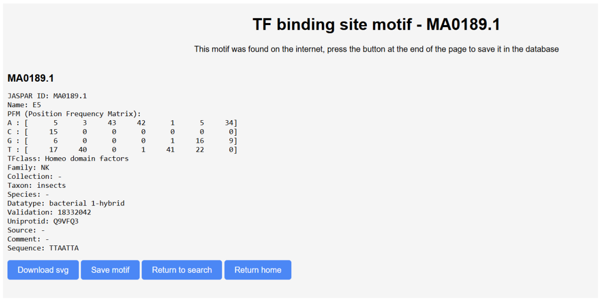
TF binding site motif - MA0139.1

MA0139.1

```
JASPAR ID: MA0139.1
Name: CTCF
PFM (Position Frequency Matrix):
A : [    87   167   281    56     8   744    40   107   851     5   333    54    12    56   104   372    82   117   402]
C : [   291   145    49   800   903    13   528   433    11     0     3    12     0     8   733    13   482   322   181]
G : [    76   414   449    21     0    65   334    48    32   903   566   504   890   775     5   507   307    73   266]
T : [   459   187   134    36     2    91    11   324    18     3     9   341     8    71    67    17    37   396    59]
TFclass: C2H2 zinc finger factors
Family: More than 3 adjacent zinc fingers
Collection: CORE
Taxon: Vertebrates
Species: 'Homo Sapiens'
Datatype: ChIP-seq
Validation: 17512414
Uniprotid: P49711
Source: -
Comment: -
Sequence: TGGCCACCAGGGGGCGCTA
```

2.5 Motif from internet page

This page is a slightly different version of the motif from database page, but for obvious reason it doesn't let the user edit or delete the motif, because the information is not stored in the database. If the user wishes to edit the motif, they first need to save it in the database through the provided button.



TF binding site motif - MA0189.1

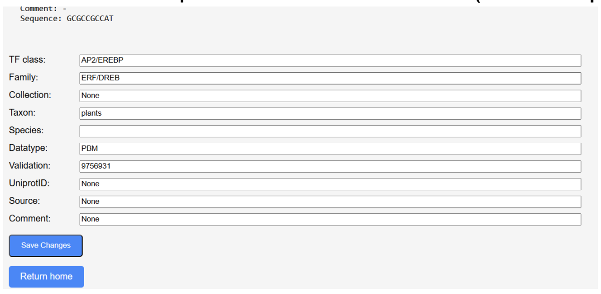
This motif was found on the internet, press the button at the end of the page to save it in the database

MA0189.1

```
JASPAR ID: MA0189.1
Name: E5
PFM (Position Frequency Matrix):
A : [     5     3    43    42     1     5    34]
C : [    15     0     0     0     0     0     0]
G : [     6     0     0     0     1    16     9]
T : [    17    40     0     1    41    22     0]
TFclass: Homeo domain factors
Family: NK
Collection: -
Taxon: insects
Species: -
Datatype: bacterial 1-hybrid
Validation: 18332042
Uniprotid: Q9VFQ3
Source: -
Comment: -
Sequence: TTAATTA
```

2.6 Edit motif

This page lets the user edits all the attributes of a motif, aside from the JASPAR ID, the name and the PFM. In the species section, more than one species can be added by simply dividing them by commas. It is advisable to not leave a space after the comma because they're going to be retained and in this way the species could be saved with a space before their name (for example, " Mus Musculus" instead of "Mus Musculus").

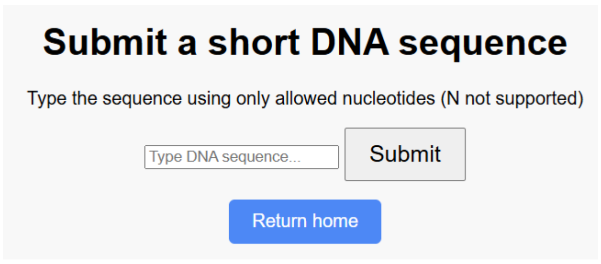


```
Comment: -
Sequence: GCGCCGCCAT
```

TF class:	AP2/EREBP
Family:	ERF/DREB
Collection:	None
Taxon:	plants
Species:	
Datatype:	PBM
Validation:	9756931
UniprotID:	None
Source:	None
Comment:	None

2.7 Provide sequence page

In this page the user can provide a nucleotide sequence: only sequences containing 'A', 'C', 'G' and 'T' are allowed.

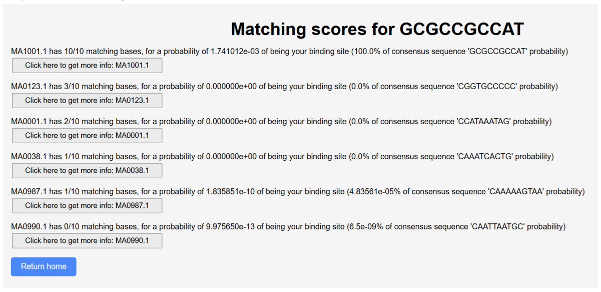


Submit a short DNA sequence

Type the sequence using only allowed nucleotides (N not supported)

2.8 Matching scores page

This page returns the matching scores obtained by comparing the provided sequence to all the motifs of the same length, using the scoring methods described previously. The user can also access each motif's information by clicking on the provided button.



Matching scores for GCGCCGCCAT

MA1001.1 has 10/10 matching bases, for a probability of 1.741012e-03 of being your binding site (100.0% of consensus sequence 'GCGCCGCCAT' probability)

MA0123.1 has 3/10 matching bases, for a probability of 0.000000e+00 of being your binding site (0.0% of consensus sequence 'CGGTGCCCCC' probability)

MA0001.1 has 2/10 matching bases, for a probability of 0.000000e+00 of being your binding site (0.0% of consensus sequence 'CCATAAATAG' probability)

MA0038.1 has 1/10 matching bases, for a probability of 0.000000e+00 of being your binding site (0.0% of consensus sequence 'CAAATCACTG' probability)

MA0987.1 has 1/10 matching bases, for a probability of 1.835851e-10 of being your binding site (4.83561e-05% of consensus sequence 'CAAAAAGTAA' probability)

MA0990.1 has 0/10 matching bases, for a probability of 9.975650e-13 of being your binding site (6.5e-09% of consensus sequence 'CAATTAATGC' probability)

Click here to get more info: MA1001.1
Click here to get more info: MA0123.1
Click here to get more info: MA0001.1
Click here to get more info: MA0038.1
Click here to get more info: MA0987.1
Click here to get more info: MA0990.1
Return home