pyJASPAR

Release v2.0.0

Aziz Khan

Sep 08, 2021

TABLE OF CONTENTS

Welcome to pyJASPAR! — a serverless interface to Biopython to query and access JASPAR motifs from different releases of JASPAR database using sqlite3.

CHAPTER

ONE

WHAT IS PYJASPAR?

pyJASPAR is a python module and a serverless interface to Biopython to query and access JASPAR motifs from different releases of JASPAR database using sqlite3.

Note: This is a serverless SQLite wrapper around the Biopython JASPAR module *Bio.motifs.jaspar.db* which requires JASPAR MySQL database sever connection details.

Currently, pyJASPAR provides access to JASPAR database releases including:

- *JASPAR2022* - http://jaspar.genereg.net/
- *JASPAR2020* - http://jaspar2020.genereg.net/
- *JASPAR2018* - http://jaspar2018.genereg.net/
- *JASPAR2016* - http://jaspar2016.genereg.net/
- *JASPAR2014* - http://jaspar2014.genereg.net/

CHAPTER

TWO

HOW TO INSTALL?

pyJASPAR is available on PyPi, through Bioconda, and source code available on GitHub. If you already have a working installation of Python, the easiest way to install the required Python modules is by installing pyJASPAR using `pip`.

If you're setting up Python for the first time, we recommend to install it using the Conda or Miniconda Python distribution. This comes with several helpful scientific and data processing libraries, and available for platforms including Windows, Mac OSX and Linux.

You can use one of the following ways to install pyJASPAR.

2.1 Install uisng Conda

We highly recommend to install pyJASPAR using Conda, this will take care of the dependencies. If you already have Conda or Miniconda installed, go ahead and use the below command.

```
conda install -c bioconda pyjaspar
```

Note: This will install all the dependencies and you are ready to use **pyJASPAR**.

2.2 Install using pip

You can install pyJASPAR from PyPi using pip.

```
pip install pyjaspar
```

Note: Make sure you're using python v3.6 or latest.

2.3 Install from source

You can install a development version by using `git` from our GitHub repository at https://github.com/asntech/pyjaspar.

```
git clone https://github.com/asntech/pyjaspar.git
cd pyjaspar
python setup.py sdist install
```

CHAPTER

THREE

HOW TO USE?

Once you have installed *pyjaspar*, you can load the module and connect to the latest release of JASPAR:

```
>>> from pyjaspar import jaspardb
```

3.1 Connect to the JASPAR

Next step is to connect to the version of JASPAR you're interested by creating a jaspardb class object. For example here we're using the the JASPAR2018.

```
>>> jdb_obj = jaspardb(release='JASPAR2018')
```

You can also check JASPAR version you are connected to using:

```
>>> print(jdb_obj.release)
JASPAR2018
```

By default it is set to latest release/version of JASPAR database. For example.

```
>>> jdb_obj = jaspardb()
>>> print(jdb_obj.release)
JASPAR2020
```

You can also connect to a local copy of JASPAR SQLite database by setting absolute path *sqlite_db_path*. For example.

```
>>> jdb_obj = jaspardb(sqlite_db_path='/path/to/jaspar.sqlite')
```

3.2 Get available releases

You can find the available releases/version of JASPAR using *get_releases* method.

```
>>> print(jdb_obj.get_releases())
['JASPAR2022', 'JASPAR2020', 'JASPAR2018', 'JASPAR2016', 'JASPAR2014']
```

3.3 Get motif by using JASPAR ID

If you want to get the motif details for a specific TF using the JASPAR ID. If you skip the version of motif, it will return the latest version.

```
>>> motif = jdb_obj.fetch_motif_by_id('MA0095.2')
```

Printing the motif will all the associated meta-information stored in the JASPAR database cluding the matric counts.

```
>>> print(motif)
    TF name YY1
    Matrix ID       MA0095.2
    Collection      CORE
    TF class        ['C2H2 zinc finger factors']
    TF family       ['More than 3 adjacent zinc finger factors']
    Species 9606
    Taxonomic group vertebrates
    Accession       ['P25490']
    Data type used  ChIP-seq
    Medline 18950698
    Matrix:
             0      1      2      3      4      5      6      7      8      9     10     ␣
↪11
    A: 1126.00 6975.00 6741.00 2506.00 7171.00   0.00  11.00  13.00 812.00 867.00 899.00␣
↪1332.00
    C: 4583.00   0.00  99.00 1117.00   0.00  12.00   0.00   0.00 5637.00 1681.00 875.00␣
↪4568.00
    G: 801.00 181.00 268.00 3282.00   0.00   0.00 7160.00 7158.00  38.00 2765.00 4655.00␣
↪391.00
    T: 661.00  15.00  63.00 266.00   0.00 7159.00   0.00   0.00 684.00 1858.00 742.00␣
↪880.00
```

Get the count matrix using *.counts*

```
>>> print(motif.counts)
        0      1      2      3      4      5      6      7      8      9     10     11
    A: 1126.00 6975.00 6741.00 2506.00 7171.00   0.00  11.00  13.00 812.00 867.00 899.00␣
↪1332.00
    C: 4583.00   0.00  99.00 1117.00   0.00  12.00   0.00   0.00 5637.00 1681.00 875.00␣
↪4568.00
    G: 801.00 181.00 268.00 3282.00   0.00   0.00 7160.00 7158.00  38.00 2765.00 4655.00␣
↪391.00
    T: 661.00  15.00  63.00 266.00   0.00 7159.00   0.00   0.00 684.00 1858.00 742.00␣
↪880.00
```

3.4 Get motifs by TF name

You can use the *fetch_motifs_by_name* function to find motifs by TF name. This method returns a list of motifs for the same TF name across taxonomic group. For example, below search will return two CTCF motifs one in vertebrates and another in plants taxon.

```
>>> motifs = jdb_obj.fetch_motifs_by_name("CTCF")
>>> print(len(motifs))
2
>>> print(motifs)
    TF name CTCF
    Matrix ID       MA0139.1
    Collection      CORE
    TF class        ['C2H2 zinc finger factors'
    TF family       ['More than 3 adjacent zinc finger factors']
    Species 9606
    Taxonomic group vertebrates
    Accession       ['P49711']
    Data type used  ChIP-seq
    Medline 17512414
    Matrix:
            0      1      2      3      4      5      6      7      8      9      10     ␣
↪11     12     13     14     15     16     17     18
    A:  87.00 167.00 281.00  56.00   8.00 744.00  40.00 107.00 851.00   5.00 333.00  54.
↪00  12.00  56.00 104.00 372.00  82.00 117.00 402.00
    C: 291.00 145.00  49.00 800.00 903.00  13.00 528.00 433.00  11.00   0.00   3.00  12.
↪00   0.00   8.00 733.00  13.00 482.00 322.00 181.00
    G:  76.00 414.00 449.00  21.00   0.00  65.00 334.00  48.00  32.00 903.00 566.00 504.
↪00 890.00 775.00   5.00 507.00 307.00  73.00 266.00
    T: 459.00 187.00 134.00  36.00   2.00  91.00  11.00 324.00  18.00   3.00   9.00 341.
↪00   8.00  71.00  67.00  17.00  37.00 396.00  59.00


    TF name CTCF
    Matrix ID       MA0531.1
    Collection      CORE
    TF class        ['C2H2 zinc finger factors']
    TF family       ['More than 3 adjacent zinc finger factors']
    Species 7227
    Taxonomic group insects
    Accession       ['Q9VS55']
    Data type used  ChIP-chip
    Medline 17616980
    Matrix:
            0      1      2      3      4      5      6      7      8      9      10     ␣
↪11     12     13     14
    A: 306.00 313.00 457.00 676.00 257.00 1534.00 202.00 987.00   2.00   0.00   2.00 124.
↪00   1.00  79.00 231.00
    C: 876.00 1147.00 383.00 784.00 714.00   1.00   0.00   0.00   4.00   0.00   0.00␣
↪1645.00   0.00 1514.00 773.00
    G: 403.00 219.00 826.00 350.00  87.00 192.00 1700.00 912.00 311.00 1902.00 1652.00  ␣
↪3.00 1807.00   8.00 144.00
    T: 317.00 223.00 236.00  92.00 844.00 175.00   0.00   3.00 1585.00   0.00 248.00 130.
↪00  94.00 301.00 754.00
```

(continues on next page)

(continued from previous page)

3.5 Search motifs based on meta-info

A more commonly used function is *fetch_motifs* helps you to get motifs which match a specified set of criteria. You can query the database based on the available meta-information in the database.

For example, here we are gettting the widely used CORE collection for vertebrates. It returns a list of 746 non-redundent motifs for JASPAR2020 release.

```
>>> motifs = jdb_obj.fetch_motifs(
collection = 'CORE',
tax_group = ['vertebrates']
)
>>> print(len(motifs))
746
```

You can loop through these motifs and perform your analysis.

```
>>> for motif in motifs:
            print(motif.matrix_id)
    MA0004.1
    MA0006.1
    -
    -
    -
    MA0528.2
    MA0609.2
```

Here is a list of meta-info *fetch_motifs* method takes as an arugment to filter the motifs.

Argument	Description
matrix_id	Takes precedence over all other selection criteria except 'all'. Only motifs with the given JASPAR matrix ID(s) are returned. A matrix ID may be specified as just a base ID or full JASPAR IDs including version number. If only a base ID is provided for specific motif(s), then just the latest version of those motif(s) are returned unless 'all_versions' is also specified.
collection	Only motifs from the specified JASPAR collection(s) are returned. NOTE - if not specified, the collection defaults to CORE for all other selection criteria except 'all' and 'matrix_id'. To apply the other selection criteria across all JASPAR collections, explicitly set collection=None.
tf_name	Only motifs with the given name(s) are returned.
tf_class	Only motifs of the given TF class(es) are returned.
tf_family	Only motifs from the given TF families are returned.
tax_group	Only motifs belonging to the given taxonomic supergroups are returned (e.g. 'vertebrates', 'insects', 'nematodes' etc.)
species	Only motifs derived from the given species are returned. Species are specified as taxonomy IDs.
data_type	Only motifs generated with the given data type (e.g. ('ChIP-seq', 'PBM', 'SELEX' etc.) are returned.
pazar_id	Only motifs with the given PAZAR TF ID are returned.
medline	Only motifs with the given medline (PubmMed IDs) are returned.
min_ic	Only motifs whose profile matrices have at least this information content (specificty) are returned.
min_length	Only motifs whose profiles are of at least this length are returned.
min_sites	Only motifs compiled from at least these many binding sites are returned.
all_versions	Unless specified, just the latest version of motifs determined by the other selection criteria are returned. Otherwise all versions of the selected motifs are returned.
all	Takes precedent of all other selection criteria. Every motif is returned. If 'all_versions' is also specified, all versions of every motif are returned, otherwise just the latest version of every motif is returned.

CHAPTER

FOUR

SUPPORT

If you have questions, or found any bug in the program, please write to us at azez.khan[at]gmail.com.

You can also report the issues to our GiHub repo

CHAPTER

FIVE

HOW TO CITE?

If you used **pyJASPAR**, please cite:

- Aziz Khan. pyJASPAR: a Pythonic interface to JASPAR transcription factor motifs. (2021). doi:10.5281/zenodo.4509415

And for the specific release of JASPAR database, please cite one of these:

JASPAR2020

- Fornes O, Castro-Mondragon JA, Khan A, et al. JASPAR 2020: update of the open-access database of transcription factor binding profiles. Nucleic Acids Res. 2020; 48(D1):D87-D92. doi: 10.1093/nar/gkz1001

JASPAR2018

- Khan A, Fornes O, Stigliani A, et al. JASPAR 2018: update of the open-access database of transcription factor binding profiles and its web framework. Nucleic Acids Res. 2018; 46:D260–D266. doi: 10.1093/nar/gkx1126

JASPAR2016

- Mathelier, A., Fornes, O., Arenillas, et al. JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. Nucleic Acids Res. 2016; 44:D110-D115.

JASPAR2014

- Mathelier, A., Zhao, X., Zhang, A. W., et al. JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. Nucleic Acids Res. 2014; 42:D142-D147.