pbtools

Release 0.1.0

Lilian Yang-crosson

Sep 23, 2019

CONTENTS

Version 0.1.0

PBTools is a program to perform analysis on molecular dynamics protein sequences encoded as Protein Blocks.

It is a simple implementation of GSATools by Pandini et al.

CHAPTER

ONE

USER MANUAL

1.1 Mutual information

The mutual information is a statistic used to compute a correlation between two random variables, and defined by the formula

$$I(A;B) = \sum_i \sum_j P(a_i, b_j) log[\frac{P(a_i, b_j)}{P(a_i)P(b_j)}]$$

where

- $I(A;B)$ is the mutual information between variable A and B,
- i and j the possible states of these variables
- $P(a_i; b_j)$ is the probability of having a_i and b_j for A and B at the same time
- $P(a_i)$ and $P(b_j)$ are the probability of having a_i in A and b_j in B respectively

To find a correlation between positions in sequences of Protein Blocks (PB) we compute the Mutual Information (MI) for each combination of position so that

- A and B represent a position in the sequence (A and B cannot be the same position)
- *i* and *j* are the values of Protein Blocks ("a" to "p")
- $P(a_i)$ and $P(b_j)$ are the probability of having a given protein block at a given position (obtained from the frequency of the PB at this position)
- The base of the logarithm is 16 (the number of PB) to normalize the values so that the maximum is 1

The number of operations rise exponentially the longer the sequence are.

1.1.1 Using PBTools to compute the MI

[1]:
```
import pbtools as pbt
print(pbt.__version__)
```

```
0.1.0
```

For the simple sequences "aaa" and "cab", we can represent it as the matrix

	0	1	2
seq1	a	a	a
seq2	c	a	b

So

$$
\begin{aligned}
I(pos0; pos1) = \\
P(a_{pos0}; a_{pos1}) \times log[\frac{P(a_{pos0}; a_{pos1})}{P(a_{pos0}) \times P(a_{pos1}}))] \\
+P(a_{pos0}; a_{pos1}) \times log[\frac{P(a_{pos0}; a_{pos1})}{P(a_{pos0}) \times P(a_{pos1}}))] \\
+P(c_{pos0}; a_{pos1}) \times log[\frac{P(c_{pos0}; a_{pos1})}{P(c_{pos0}) \times P(a_{pos1}}))] \\
+P(c_{pos0}; a_{pos1}) \times log[\frac{P(c_{pos0}; a_{pos1})}{P(c_{pos0}) \times P(a_{pos1}}))] \\
= 0.5 \times log(\frac{0.5}{0.5}) \\
= 0 = I(pos2; pos1)
\end{aligned}
$$

And

$$
\begin{aligned}
I(pos0; pos2) = \\
P(a_{pos0}; a_{pos2}) \times log[\frac{P(a_{pos0}; a_{pos2})}{P(a_{pos0}) \times P(a_{pos2}}))] \\
+P(a_{pos0}; b_{pos2}) \times log[\frac{P(a_{pos0}; b_{pos2})}{P(a_{pos0}) \times P(b_{pos2}}))] \\
+P(c_{pos0}; a_{pos2}) \times log[\frac{P(c_{pos0}; a_{pos2})}{P(c_{pos0}) \times P(a_{pos2}}))] \\
+P(c_{pos0}; b_{pos2}) \times log[\frac{P(c_{pos0}; a_{pos2})}{P(c_{pos0}) \times P(b_{pos2}}))] \\
= 0.5 \times log(\frac{0.5}{0.5 \times 0.5}) \times 2 \\
= 0.5 \times 0.25 * 2 \\
= 0.25
\end{aligned}
$$

[2]:
```
pbt.mutual_information_matrix(["aaa", "cab"])
```

[2]:
```
array([[0.  , 0.  , 0.25],
       [0.  , 0.  , 0.  ],
       [0.  , 0.  , 0.  ]])
```

We can observe that at position 0,2 the MI is 0.25 just as calculated earlier, so the matrix computed by PBTools is correct.

1.2 Using the API

[1]:
```
import matplotlib.pyplot as plt
import networkx as nx
import pbxplore as pbx

import pbtools as pbt

print(f"networkx version {nx.__version__}")
```

(continues on next page)

(continued from previous page)

```
print(f"pbxplore version {pbx.__version__}")
print(f"pbtools version {pbt.__version__}")

%matplotlib inline

# Bigger figures for the whole notebook.
plt.rcParams['figure.figsize'] = [30, 20]
```

```
networkx version 2.3
pbxplore version 1.3.8
pbtools version 0.1.0
```

[2]:
```
# Get trajectory and topology files.
! bash ../data/small_run.sh
```

```
-2019-09-18 17:30:01-  https://raw.githubusercontent.com/pierrepo/PBxplore/master/
↪demo_doc/psi_md_traj.gro
Résolution de ewebproxy.univ-paris-diderot.fr (ewebproxy.univ-paris-diderot.fr)... 81.
↪194.35.225, 2001:660:3301:80fb::225
Connexion à ewebproxy.univ-paris-diderot.fr (ewebproxy.univ-paris-diderot.fr)|81.194.
↪35.225|:3128... connecté.
requête Proxy transmise, en attente de la réponse... 200 OK
Taille : 18630 (18K) [text/plain]
Enregistre : «psi_md_traj.gro.2»

psi_md_traj.gro.2   100%[===================>]  18,19K  -.-KB/s    ds 0s

2019-09-18 17:30:01 (51,2 MB/s) - «psi_md_traj.gro.2» enregistré [18630/18630]

-2019-09-18 17:30:01-  https://raw.githubusercontent.com/pierrepo/PBxplore/master/
↪demo_doc/psi_md_traj.xtc
Résolution de ewebproxy.univ-paris-diderot.fr (ewebproxy.univ-paris-diderot.fr)... 81.
↪194.35.225, 2001:660:3301:80fb::225
Connexion à ewebproxy.univ-paris-diderot.fr (ewebproxy.univ-paris-diderot.fr)|81.194.
↪35.225|:3128... connecté.
requête Proxy transmise, en attente de la réponse... 200 OK
Taille : 378124 (369K) [application/octet-stream]
Enregistre : «psi_md_traj.xtc.2»

psi_md_traj.xtc.2   100%[===================>] 369,26K  -.-KB/s    ds 0,02s

2019-09-18 17:30:02 (22,8 MB/s) - «psi_md_traj.xtc.2» enregistré [378124/378124]
```

1.2.1 Assigning the PBs and creating the MI matrix

[3]:
```
# Reading the files and assigning Protein Blocks.
trajectory = "psi_md_traj.xtc"
topology = "psi_md_traj.gro"

chains = pbx.chains_from_trajectory(trajectory, topology)

all_sequences = []
for comment, chain in chains:
    dihedrals = chain.get_phi_psi_angles()
```

(continues on next page)

(continued from previous page)

```
    sequence = pbx.assign(dihedrals)
    all_sequences.append(sequence)
```

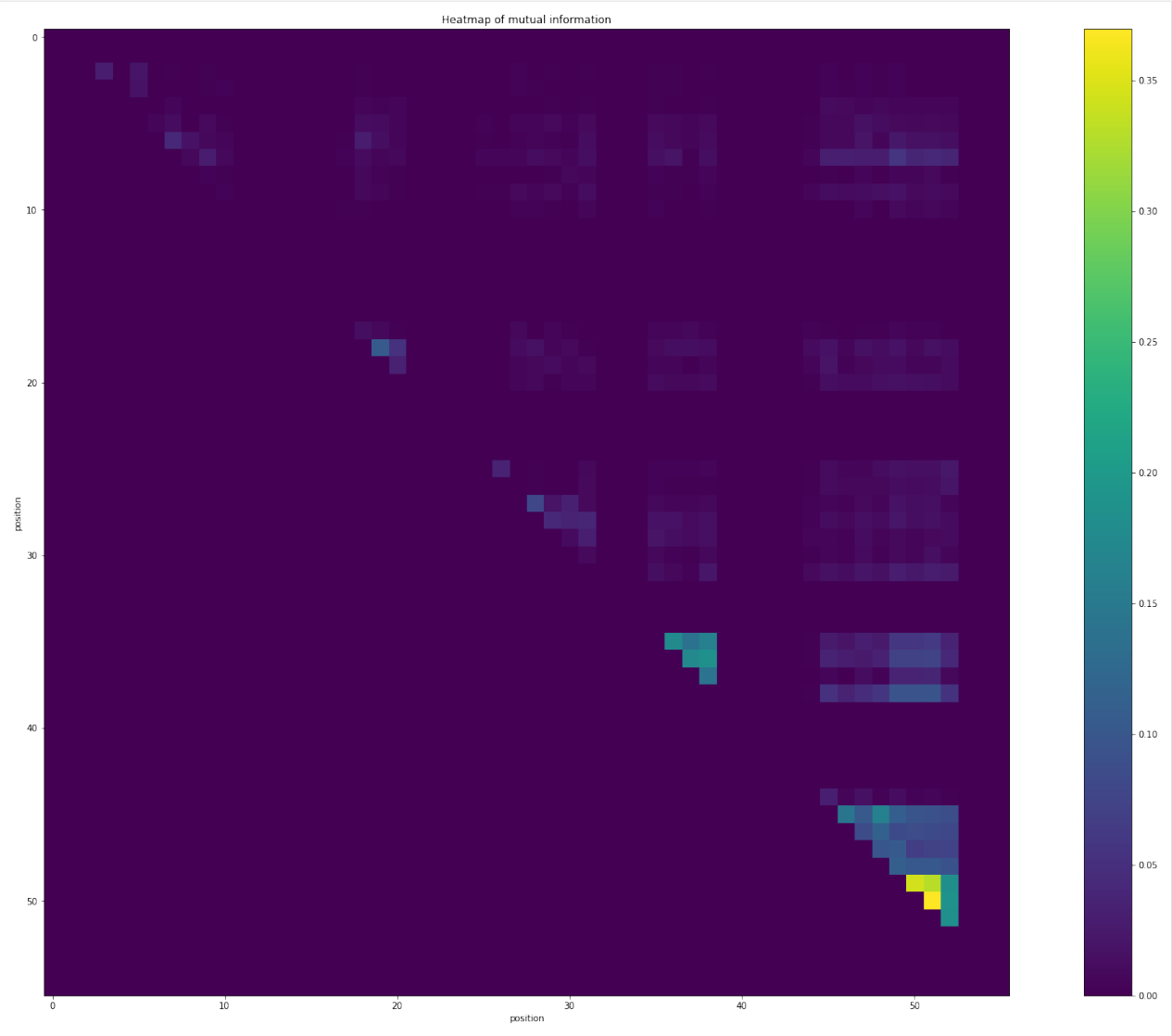
```
Frame 1/225.
Frame 100/225.
Frame 200/225.
Frame 225/225.
```

[4]:
```
# Creating the matrix, might be slow.
MI_matrix = pbt.mutual_information_matrix(all_sequences)
print("Done")
```

```
Done
```

[5]:
```
# Getting the heatmap of mutual information.
plt.title("Heatmap of mutual information")
plt.xlabel("position")
plt.ylabel("position")

plt.imshow(MI_matrix)
plt.colorbar()
```

[5]:
```
<matplotlib.colorbar.Colorbar at 0x7fb16400b990>
```

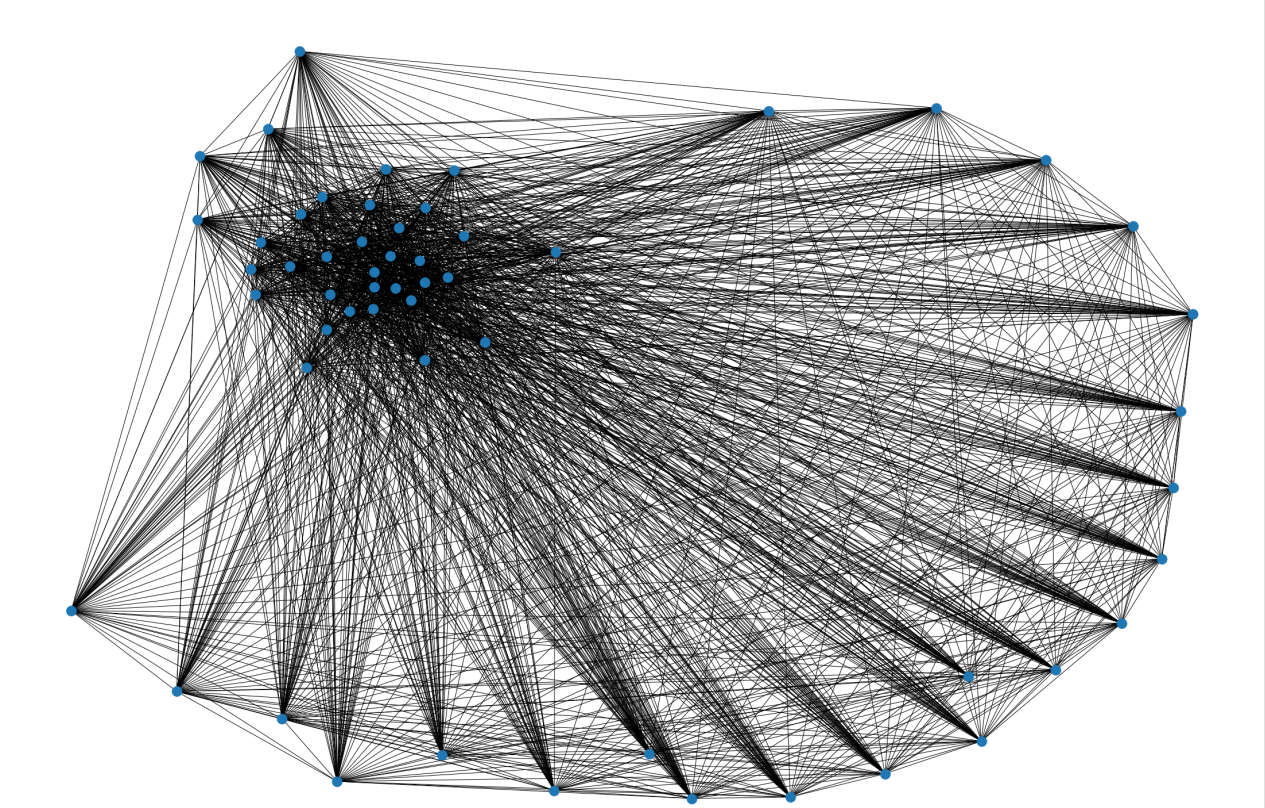


1.2.2 Network manipulation

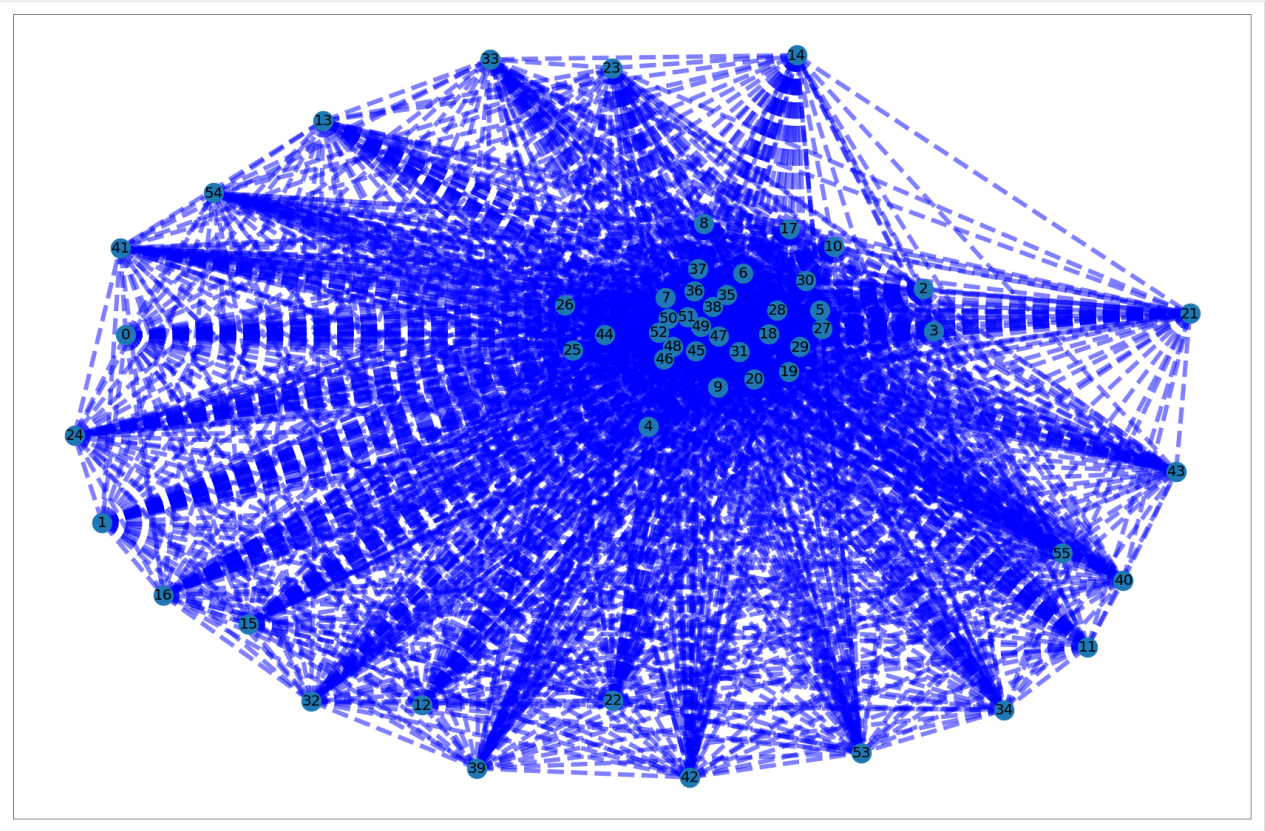
[6]:
```
# Creating the network.
PB_graph = pbt.interaction_graph(MI_matrix)
```

[7]:
```
nx.draw(PB_graph)
```

```
/home/sdv/m2bi/lyang_crosson/.conda/envs/PBTools_dev/lib/python3.7/site-packages/
↪networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in 3.3.␣
↪Use np.iterable instead.
  if not cb.iterable(width):
```



[8]:
```
# Differentiating edges by weight, adapted from :
# https://networkx.github.io/documentation/stable/auto_examples/drawing/plot_weighted_
↪graph.html

elarge = [(u, v) for (u, v, d) in PB_graph.edges(data=True) if d['weight'] > 0.5]
esmall = [(u, v) for (u, v, d) in PB_graph.edges(data=True) if d['weight'] <= 0.5]

pos = nx.spring_layout(PB_graph)  # positions for all nodes

# nodes
nx.draw_networkx_nodes(PB_graph, pos, node_size=700)

# edges
nx.draw_networkx_edges(PB_graph, pos, edgelist=elarge,
                       width=6)
nx.draw_networkx_edges(PB_graph, pos, edgelist=esmall,
                       width=6, alpha=0.5, edge_color='b', style='dashed')

# labels
nx.draw_networkx_labels(PB_graph, pos, font_size=20, font_family='sans-serif')
plt.show()
```


33
23
14
13
64
41
8
17
10
37
6
30
36
35
7
38
28
5
2
21
26
50
51
49
27
3
0
44
52
47
18
29
25
48
45
31
46
20
19
9
4
24
43
1
55
40
16
15
11
32
12
22
34
39
42
53

CHAPTER

TWO

REFERENCE MANUAL

2.1 pbtools

Perform analysis on protein sequences encoded as Protein Blocks

Provide functions to compute a Mutual Information matrix from sequences.

pbtools.**interaction_graph**(*matrix*)
Create a networkx graph object from a (square) matrix.

Parameters **matrix** (*numpy.ndarray*) –

Matrix of mutual information, the information for the edges is taken from the upper matrix

Returns **graph** – The graph with MI as weighted edges and positions as nodes

Return type networkx.Graph()

Raises **AssertionError** – If the matrix is not square

pbtools.**mutual_information**(*pos1*, *pos2*)
Computes the Mutual Information from 2 Series of Protein Blocks.

Parameters

- **pos1** (*pandas.Series*) – Protein Blocks of a given sequence position.
- **pos2** (*pandas.Series*) – Protein Blocks of a given sequence position.

Returns **MI** – Mutual Information of the 2 positions.

Return type float

Raises **AssertionError** – If Series are not provided or have different lengths.

pbtools.**mutual_information_matrix**(*sequences*)
Return the matrix of Mutual Information for all positions.

Parameters **sequences** (*list of str*) – Sequences obtained from the file(s)

Returns **MI_matrix** – Matrix of Mutual Informations.

Return type numpy.ndarray

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

p

INDEX

I

M

P