TCRcluster Documentation

Group 1

1 What is TCRcluster?

TCRcluster is a clustering tool to calculate distances of T-cell receptor (TCR) sequences based on the CDR3 regions and then cluster them using distances. The results of TCR clustering will be verified by epitope sequences they recognize. Furthermore, this clustering outcome can be used to facilitate the classification of new TCR sequences in the future.

2 Key algorithms behind this software

2.1 Distance Calculation

Dynamic programming is used to calculate the similarity (distance) between each TCR sequence. The diversity of CDR3α and CDR3β sequences represent the variety of TCR sequences. Based on each CDR3 sequence, BLOSUM-62 or BLOSUM-80 matrix is used to assign points for matched or mismatched amino acids according to all other sequences, while two forms of penalties (gap open & gap extend) are employed to deal with different gaps.

2.2 Clustering

In clustering, we develop a graph clustering algorithm referring to the Louvain algorithm. The "modularity" is used to show the entire quality of how the network is segmented by communities (clusters). By constantly moving nodes to the neighbor's community, the modularity can be optimized to the current best level. The overall process contains two steps that are modularity optimization and aggregation. First, every node in the network tries to merge into the community of its neighbor, which brings about a change in the modularity (ΔQ). For each node, move the node into the corresponding community with the biggest positive ΔQ. Iterate this step constantly until there is no positive ΔQ change for any node in a round. In the second step, the nodes in the same community will be aggregated into a new super-node. A new network is built for another run of the first step and the whole cycle reruns. Finally, the whole process stops until there is no change in the first step.

2.3 Visualization

Firstly, an undirected graph is created to store the CDR3 sequences as nodes and TCR distances as edges. Then, the coordinates of the nodes in the two-dimensional canvas can be calculated using the Fruchterman-Reingold (FR) algorithm. The main idea of this FR algorithm is to treat all nodes as charged particles. In this way, there are two forces act on each node: the repulsive coulomb force and the gravitational tensile force that promotes nodes with high similarity to get closer to each other. Then the velocity and motion of the nodes can be calculated in each time unit. After continuous iterative calculation, a dynamic equilibrium state is finally achieved. The final positions of nodes are then used for assessing the clustering.

3 Running environment & reliable packages

The running environment of this software is ***Python 3.7***. Some other packages need to be downloaded prior to using this software, including ***numpy (v1.23.3)***, ***pandas (v1.4.4)***, ***networkx (v.2.8,6)***, and ***matplotlib (v.3.5.3)***. We provide a *requirements.txt* file, which can be used to download all needed packages. Just run this command:

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

4 Parameter explanation (including input & output formats)

4.1 Input format

A .csv file must contain at least one column with the fixed name: **CDR3b**, which represents CDR3β sequences. Notably, the input file must also contain the **CDR3a** column representing CDR3α sequences if you want to consider the influence of CDR3α (Use **-w/–weight**, see below for more information). Additionally, **peptide** column representing the sequences of corresponding epitope amino acid must be contained if you want to verify the result of clustering (Use **-ver/–verification**, see below for more information). The software will raise information if the input format is wrong.

4.2 Parameter explanation

General usage: TCRcluster.py -i [-t] [-B] [-e] [-o] [-w] [-s] [-ver] -out
Arguments:
-h, –help: Show the quick and brief help message.
-i, –input: The path of the input .csv file.
-t, –trim: If this parameter is added, the conserved sequences of CDR3 (2 amino acids on the N-terminal side and the C-terminal side) will be trimmed and not used when calculating distances. If your input file has already contained trimmed sequences, please do not set this parameter.
-B, –BLOSUM: Select the BLOSUM matrix used in global alignment to assign points for matched or mismatched amino acids. We currently provide two matrixes: BLOSUM-62 and BLOSUM-80. You can choose either 62 or 80 for this value. If your provided sequences seem to have higher similarities, we suggest using -B 80 or –BLOSUM 80. Additionally, if you have no idea about the degree of similarities of your sequences, just use our default value: -B 62 or –BLOSUM 62.
-e, –extend: The penalty points for a gap extend (default -1).
-o, –open: The penalty points for a gap open (default -5).
-w, –weight: The percentage of CDR3b weight when calculating the distances between each TCR sequence. This value must be integers from 0 to 100 (default 100). For example, “100” means that only the CDR3b sequence will be considered, while “90” means that the final distances between each TCR sequence are equal to 90% of the CDR3b sequence distances plus 10% of the CDR3a sequence distances. We highly recommend setting more than 90 for this parameter.
-s, –select: Only the top percent of distances from each TCR sequence to others will be considered when clustering. This value must be integers from 1 to 50 (default 10). For instance, “10” means that for each CDR3 sequence, only the top 10% distances to others will be preserved for clustering. A bigger percentage may conclude a more convincing conclusion but with a much longer running time (we highly recommend using the default value for this parameter).
-ver, –verification: Whether to verify the result of TCR clustering using the corresponding epitope sequences. If this parameter is added, a .csv file containing the consensus motif of each cluster will be output. Remember to add the **peptide** column in your input file if this parameter is set.
-out, –output: The path and the title name for output files. We have more than one output file, therefore you just need to provide the title name, not including the format of the output files. For example, use -out test instead of -out test.txt or test.png.

4.3 Output format

The software will deliver two output files if you do not set **-ver**, while three files if you set it. The first is a .txt file showing the result of clustering. Numbers in each community represent the original rows (starting from 0) in your input file, meaning that the TCRs in these rows are clustered into the same community. The second is a .png file showing a clustering plot for visualization. The third is a .csv file (only when **-ver** is set) containing the consensus motif of epitope antigens in each cluster, together with the information of epitope antigen and CDR3b sequences for each TCR. You can see the below examples for a more intuitive sense.

4.4 Additional information

The software will show the step it is currently running and you can choose to stop running it during visualization since this step costs much time. In this situation, you will not get the .png output file. The row number (number of TCR sequences) is suggested to be more than 100 since fewer data will lead to a relatively bad clustering result and visualization. Due to the randomization of clustering and visualization steps, you can run this software with the same parameters and input file more than one time to get a satisfactory result.

5 Examples to use this software

Two provided input test files **(test1.csv & test2.csv)** contain experimental validated TCR-epitope pairs including antigen peptides and CDR3 sequences. You can run the following commands to test (test files and all .py files should be in the same folder). The results may be different from the examples due to the randomization of our algorithms.

5.1 Example 1: 125 TCRs

```
TCRcluster.py -i test1.csv -B 80 -ver -out output1
```

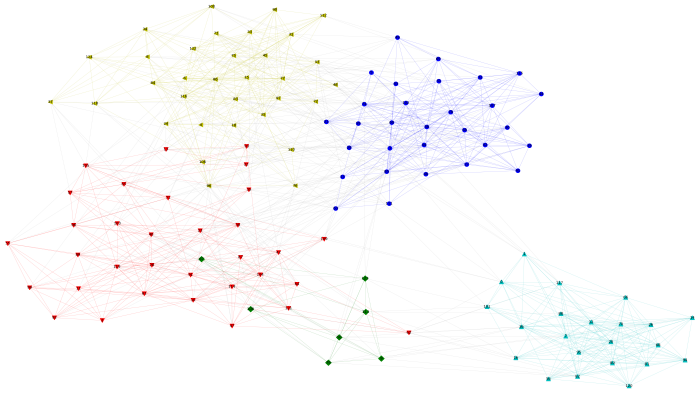

Figure 1: The visualization of clustering for test1.csv **(output1.png)**

5.2 Example 2: 300 TCRs

```
TCRcluster.py -i test2.csv -t -B 62 -w 90 -ver -out output2
```

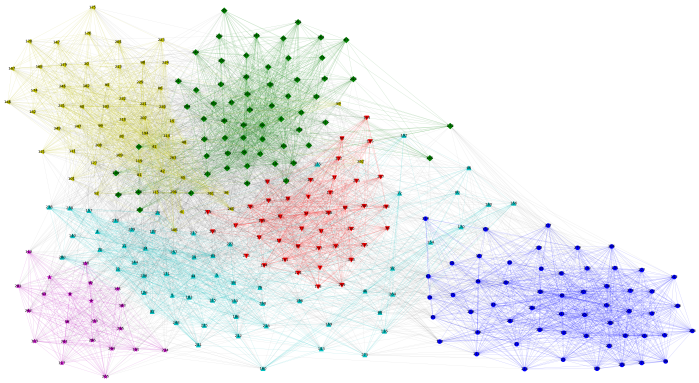

Figure 2: The visualization of clustering for test2.csv **(output2.png)**

You can see **output.txt** for clustering information, **output.png** for clustering visualization and **output.csv** for verification information of both input test files. These output files are provided by us.