
# User Guide for ProCon-Python (Protein Conservation) Version 1.0

---

© 2005 - 2020

School of Computer Science and Technology, Soochow University, Suzhou, China

and

Institute of Biomedical Technology, University of Tampere, Finland

# Catalogue

# Introduction

ProCon is tool for locating and visualization of evolutionary conservation in protein sequences. The method can identify three types of conservation, namely identity (type I), physicochemical similarity (II), and covariant conservation (III). The conservative sites of type I and II are located with entropy calculation and the third type is identified by calculation of mutual information. The interacting networks formed by covariant pairs can also be identified. All the three types of conservation can be visualized in a representative protein structure. The tool performs exhaustive analysis results of which can be used e.g. for identifying different types of conservative residues, studying protein-protein interactions, explaining consequences of disease-causing mutations and mutant design for protein engineering.

# 0 Setting up

## Environment requirement

The ProCon package is implemented in Python3. In order to properly use the library, please make sure that your project's Python version is later than 3.5. (You may download it from https://www.python.org/downloads/)

The library can be run on both Mac OS X, Windows and Linux.

## Online installation

The easiest way to install ProCon is to use pip. Please make sure your pip version is the latest one before you start to install. You can upgrade your pip by using the following command.

```
python -m pip install --upgrade pip
```

To install ProCon run:

```
pip install ProCon==0.0.1
```

## Offline installation

Normally, we use `pip install` to install ProCon online. But sometimes we may need to use `.tar.gz` file to install the library. You can refer to the following steps to complete the installation.

1. Download the `.tar.gz` file

   Link: https://pypi.org/project/myProCon/0.0.1/

2. Install wheel model

   In command-line window, run:

   ```
   pip install wheel
   ```

3. Install `.tar.gz` file

   In command-line window, use the command `cd` to jump to the directory of the `ProCon-0.0.1.tar.gz` file run:

   ```
   python3 setup.py install ProCon-0.0.1.tar.gz
   ```

# 1 Calculate namely identity (type I)

## Import a FASTA-format sequence file

The main class of calculating namely identity (type I) is called `ProbabilityCalculator`. Find the route of your FASTA-format sequence file run:

```
pc = ProbabilityCalculator.ProbabilityCalculator("file
route")
```

to import the sequence file. The system calculates the results according to the default parameter values. These parameters can also be modified.

## Configure parameters

1. **Set gap percent**

   The second parameter of the `ProbabilityCalculator` is gap percent. Its default value is 0.1 and if you want to modify it (For example 0.2) run:

   ```
   pc =
   ProbabilityCalculator.ProbabilityCalculator("file
   routr", 0.2)
   ```

2. **Set p values**

   The third and fourth parameters of the `ProbabilityCalculator` is p-value 1 and p-value 2. Their default value are 0.01 and 0.05. If you want to modify them (For example 0.01 and 0.1) run:

   ```
   pc =
   ProbabilityCalculator.ProbabilityCalculator("file
   routr", 0.2, 0.01, 0.1)
   ```

## Get the results

The default setting calculates the sequence by classifying it into 20 groups (20 + 'gap'').

1. **Achieve raw result**

   run

   ```
   pc.get_entropy_0()
   ```

2. **Print sorted result**

   run

   ```
   pc.print_sorted_inf_20()
   ```

   Please notice that the results, few of which have eliminated because their gap percent is over the defined value.

3. **Export sorted result to file**

   run

   ```
   pc.inf_to_file_20("file name")
   ```

   If the file does not exist, the method will create a new one. If the file exists, the method will overwrite it. So please be cautious that if you have a file named the same name before using this method.

   Example:

| rank | position | information |
|---|---|---|
| 1 | 324W | 2.901 |
| 2 | 5S | 2.32 |
| 3 | 48L | 2.112 |
| 4 | 240F | 1.968 |
| 5 | 328L | 1.836 |
| 6 | 7F | 1.815 |
| 7 | 60L | 1.807 |
| 8 | 145V | 1.69 |
| 9 | 242V | 1.644 |
| 10 | 312V | 1.464 |
| 11 | 316T | 1.371 |
| 12 | 143I | 1.292 |
| 13 | 45L | 1.226 |
| 14 | 314S | 1.209 |
| 15 | 44R | 1.141 |

## Some useful methods

1. `.get_seq_number()`

Return the number of sequences in imported FASTA-format sequence file.

2. `.get_res_number()`

   Return the number of residue in a single sequence.

3. `.get_seq()`

   Return the sequences in the sequence file.

4. `.get_seq_names()`

   Return the names of the sequences in the sequence file.

5. `.display_seq()`

   Display all sequences in the terminal.

# 2 Calculate physicochemical similarity (type II)

---

Physicochemical similarity (type II) is calculated based on the namely identity (type I).

## Configure parameters

1. **Set gap percent and p values**

   If you want the change the parameters such as gap percent, p-value 1 and p-value 2 when calculating physicochemical similarity, you need the use the object instantiated from class `ProbabilityCalculator` to configure the parameters **before** your initialization.

   Use `.set_gap_percent(p)` to configure the gap percent.

   Use `.set_p_value_1(p_1)` to configure the p-value 1.

   Use `.set_p_value_2(p_2)` to configure the p-value 2.

2. **Set amino acid groups**

   a. We provide several predesigned groups in the library.

   i. 6 groups (**default**)

   run

   ```
   .set_default_group()
   ```

   The six groups are named as

   **hydrophobic:** C, F, I, L, M, V, W, Y

   **AT:** A, T

   **negative:** D, E

   **conformational:** G, P

   **polar:** N, Q, S

   **positive:** H, K, R

ii. 4 groups

run

```
.set_group_4()
```

The four groups are named as

**non polar:** A, F, G, I, L, M, P, V, W

**uncharged polar:** C, N, Q, S, T, Y

**acidic:** D, E

**basic:** H, K, R

iii. 3 groups

run

```
.set_group_3()
```

The three groups are named as

**essential:** F, H, I, K, L, M, T, V, W

**non-essential:** A, D, E, N, S

**conditionally essential:** C, G, P, Q, R, Y

iv. 7 groups

run

```
.set_group_7()
```

The seven groups are named as

**aliphatic:** A, G, I, L, P, V

**aromatic:** F, W, Y

**acidic:** D, E

**basic:** H, K, R

**hydroxylic:** S, T

**sulfur-containing:** C, M

**amidic:** N, Q

b. The library also enables you to set your own amino acid groups.

Firstly, the number of the groups should be set. Use the method `.set_number_of_group(number)`, for example

```
pc.set_number_of_group(6)
```

Secondly, you have to set the groups' names. Use the method `.set_group_names(group_names)`, for example

```
pc.set_group_names(["hydrophobic", "AT",
"negative", "conformational", "polar",
"positive", "gap"])
```

Then, a list that represent the group should be set. Use the method `.set_group(group)` and the corresponding list is

```
['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S',
'T', 'V', 'W', 'Y', '-' ]
```

For example

```
pc.set_group([1, 0, 2, 2, 0, 3, 5, 0, 5,
0, 0, 4, 3, 4, 5, 4, 1, 0, 0, 0, 6])
```

To do so, amino acid 'A' will be classified to group 1, 'C' will be classified to group 0, 'D' will be classified to group 2 and so on.

At last, the module needs to recalculate the result. **(*IMPORTANT*. If not to do so, the result will still remain the same as the default one)** Use the method `.re_calculate()`

```
pc.re_calculate()
```

to update the result.

## Initialization

The main class of calculating the physicochemical similarity (type II) is `MutualInformation`.

To initialize, run:

```
mi = MutualInformation.MutualInformation(pc)
```

## Get the results

1. **Achieve raw result**

   run

   ```
   mi.get_mut_inf()
   ```

2. **Display sorted result**

run

```
mi.print_mutu_info()
```

3. **Export sorted result to file**

   run

   ```
   mi.mut_to_file("file name")
   ```

   The result is a two-dimensional matrix divided by ','. If the file does not exist, the method will create a new one. If the file exists, the method will overwrite it. So please be cautious that if you have a file named the same name before using this method.

   Example:

| Rank | Site1 | Site2 | Mutual Information |
|---|---|---|---|
| 1 | 6I | 41F | 50.198 |
| 2 | 2I | 8L | 35.079 |
| 3 | 49T | 61S | 27.664 |
| 4 | 44R | 46F | 26.007 |
| 5 | 41F | 45L | 25.004 |
| 6 | 45L | 63Y | 22.616 |
| 7 | 41F | 63Y | 21.340 |
| 8 | 311Y | 321R | 19.804 |
| 9 | 9K | 141G | 19.054 |
| 10 | 153E | 321R | 18.642 |
| 11 | 6I | 63Y | 18.619 |
| 12 | 49T | 137G | 18.480 |
| 13 | 247G | 332I | 18.085 |
| 14 | 9K | 11S | 17.863 |
| 15 | 41F | 43K | 17.829 |
| 16 | 43K | 45L | 17.797 |
| 17 | 139K | 140K | 17.621 |

# 4 Calculate covariant conservation (type III)

## Initialization

***The default group of the amino acid is 20 + 1.*** If you want to reset the group, see *Configure parameters* in type II.

The main calss of calculating the covariant conservation (type III) is `TripletFinder`.

To initialize, run:

```
tp = TripletFinder.TripletFinder(mi)
```

## Get the results

1. Achieve raw result

   run

   ```
   tp.get_triplets()
   ```

2. Display result

   run

   ```
   tp.display_triplets()
   ```

3. Export result to file

   run

   ```
   tp.tps_to_file("file name")
   ```

   If the file does not exist, the method will create a new one. If the file exists, the method will overwrite it. So please be cautious that if you have a file named the same name before using this method.

   Example:

   | rank | site1 | site2 | site3 |
   |---|---|---|---|
   | 1 | 43E | 45K | 50A |
   | 2 | 43E | 45K | 76S |
   | 3 | 43E | 45K | 77S |
   | 4 | 43E | 50A | 76S |
   | 5 | 43E | 50A | 77S |
   | 6 | 43E | 76S | 77S |
   | 7 | 45K | 50A | 76S |
   | 8 | 45K | 50A | 77S |
   | 9 | 45K | 76S | 77S |
   | 10 | 49P | 50A | 76S |
   | 11 | 50A | 55G | 58P |
   | 12 | 50A | 55G | 77S |
   | 13 | 50A | 58P | 76S |

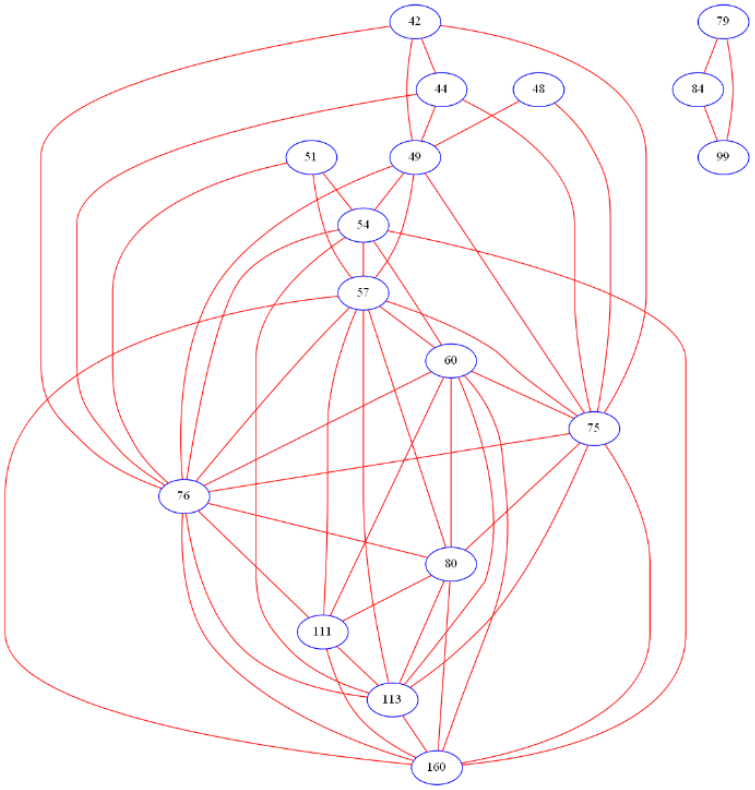
4. Generate graph

   run

   ```
   tp.show_graph("file name")
   ```

   If the file does not exist, the method will create a new one. If the file exists, the method will overwrite it. So please be cautious that if you have a file named the same name before using this method.

   Example:




# 5 About us

The tool is developed by:

Yang Yang, Neng Qian, Guanchen Zhu, André Norrgård, Mats Aspnäs, Jan Westerholm, Mauno Vihinen and Bairong Shen.

***Email:*** 1729401199@stu.suda.edu.cn、yyang@suda.edu.cn

© 2005 - 2020

School of Computer Science and Technology, Soochow University, Suzhou, China

and

Institute of Biomedical Technology, University of Tampere, Finland