

# Use EPTA

## Before you start

---

### Configuration file

---

A configuration file named 'Config.ini' can be found under the root path of the EPTA folder, which stores all default command-related parameters. Users can edit it manually to change the default settings of the tool.

Additionally, EPTA generates a 'Config.ini' file in the input file path after a run. Thus, if one needs to run EPTA multiple times using the same input file, edit configuration file in the input file path is also a option.

### Log file

---

EPTA automatically generates a log file that records everything that happened during a run. Users can find the log under the output path of each run.

Also, a configuration log will be automatically generated in the output path that allows user check or confirm configuration of a run.

### Play with EPTA

---

Not sure how to make it.

## Make a tree from FASTA file

---

An annotated phylogenetic tree can be built directly from FASTA file, and able to add protein ID, description, organism lineage, protein domain are able to be added to the tree.

### Example command-line

---

Under construction

### Input file

---

A '-i' flag is what we used to declare input file, which is obligatorily required. The inputted file can be a FASTA file in plain text format, or contained in a gunzip or tar.gz compressed file, or a folder contains any the aforementioned file types.

Input a FASTA file:

```
epta -i test_fasta.fasta -o ./test
```

Input a gunzip file:

```
epta -i test_fasta.gz -o ./test
```

Input a tar.gz file:

```
epta -i test_fasta.tar.gz -o ./test
```

Input a folder:

```
-ls ./fasta_files  
'test_fasta.fasta' 'test_fasta.gz' 'test_fasta.tar.gz'  
  
epta -i ./fasta_files -o ./test
```

## Output path

A '-i' flag is what we used to store output file, which is obligatory required.

Our tool will automatically detect whether the output path exists and if it is, you will see the following caution:

```
Output directory is existed, do you want to continue? (Y/N)
```

### Caution

Once the user input **Y** and **press enter**, all tool-related files, including the previous dataframe, parsed fasta file, etc., will be **removed** from that path.

If you would like to skip manual confirmation, add **-redo** flag in your command line.

## Check point

EPTA will automatically generates a check point after it starting a run. You will see the following caution if any check point file has been found in the output path:

```
Output directory is existed, do you want to read check point? (Y/N)
```

To continue the run from the place where last run ended, input **Y** and press enter.

Same as output path check, if you would like to skip manual confirmation, add **-redo** flag in your command line.

## Parse FASTA

### Sequence recognition

We introduced Biopython module to distinguish sequence info. By default, protein name and ID will be extracted from sequence headers automatically. Users do not need any command-line flags to make it happen.

### Dataframe

Our tool will generate a human-readable dataframe called 'info\_index.tsv' in tsv format under the output path after parse all inputted files. Furthermore, if users choose to run Pfamscan, an independent tsv file named 'Pfamscan\_result.tsv' will be generated under the output path in order to inspect all domain-related data.

For a line in the dataframe file, index is a unique random ID assigned by EPTA, and columns are arranged as:

Random ID	Header	ID	Name	Organism Lineage	Domain Information[database]	Domain Information[Pfamscan]	Pfamscan JSON data
--------------	--------	----	------	---------------------	---------------------------------	---------------------------------	--------------------------

If a parameter is not requested, the following columns will move forward in order.

### Duplicate headers?

If you want to keep all duplicate sequence in your file, add **-dh** command to your command line.

### Local data?

Local sequences that contained in FASTA file need to be marked with a '**lcl**' identifier, in order to extract information from it **automatically**. Protein name, ID, and taxonomy can be directly included in a header with specific identifiers and splited with a vertical bar ('|'), while the order of their **arrangements doesn't matter**. And, Pfamscan data should be included in another file if you do not want to fetch domain information or run Pfamscan by EPTA, and the format of domain information should be in a JSON format which is the same with Pfamscan's result.

Parameter identifiers:

Parameter	Identifiers
Name	[NAME]
ID	[ID]
Taxonomy	[TAXON]

Example of a local sequence:

```
1 >lcl| [ID]]OAO11745.1 | [NMAE]hydrogenase | [TAXON] cellular organism
2 MLSRLSRIATTKSMLVMNAARSFAAEAQGLVSVKINGNEYKVPEGMTVLEACQAQGIHVPFVCHHPRLI
```

Example of a domain information format:

```
1 >lcl| [ID]]OAO11745.1
2 [{"model_length":"82","align":["#HMM          pvtltfDGkevtvpeGdtvasAllan
```

#### Hint

A local sequence tag and a [ID] block is the minimum requirement for EPTA to recognize a sequence.

### Taxonomy information

When using a '**-tax**' flags to enable the taxonomy finding functionality, EPTA can fetch organism lineage automatically from the NCBI database with a given protein ID, and the taxonomy information will be stored in dataframe in the following format:

```
cellular organisms; Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Opali
```

Once a sequence has organism lineage, it would be easy for show species or select proteins from organism under a certain classification in later process.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax
```

### Protein name from database

EPTA can search protein name according to accession numbers from Entrez database. To enable this function, add **-name** flag to the command line.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name
```

### Run PfamScan

EPTA introduced PfamScan to allow users annotate protein domains on the tree.

Add **-dom** flag to the command line to enable domain annotating.

Add **-pfam** flag to the command line to run PfamScan to search domain information.

Pfamscan search can provide name, accession number, hit sequence, envelope, e-value, bit score, and active site of a protein domain. Which is a recommended way to obtain protein domain information.

Alongside **-pfam** flag, **-pev [E-value]** and **\*\*'-pas'\*\*-** can be used to specify a e-value and to enable active site functionality in Pfamscan search. Meanwhile, both shortened domain data in the dataframe and a human-readable tsv file will be automatically generated after a Pfamscan search in order to provide detailed information of protein domains.

We also introduced the **-em** flag for users to provide an email address for receive protein messages from web services (for example, you are temporarily banned).

Example of Pfamscan result in dataframe:

```
{'Iron only hydrogenase large subunit, C-terminal domain; PF02906.14; eval
```

Example of Pfamscan details form:

	seq_name	seq_id	alignment_start	alignment_end	env
i6j8150wKOrP7Xpz	GIQ79514.1	GIQ79514.1	1	186	1
i6j8150wKOrP7Xpz	GIQ79514.1	GIQ79514.1	195	249	194
i6j8150wKOrP7Xpz	GIQ79514.1	GIQ79514.1	276	415	276

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -pev 10 -pas -em tes
```

Caution

According to the rule of the Pfamscan web service, one can submit at most 3000 sequences in a maximum of 30 batch jobs when using the Lite version.

Therefore, if you have more than 1500 sequences that need to run Pfamscan search, please do not run our tool again immediately after keyboard interrupting a run that has already submit sequences to the server.

## Multiple sequence alignment

EPTA introduced MAFFT and MUSCLE as multiple sequence alignment programs. The default program is MAFFT, which could be changed in the configuration file.

After run multiple sequence alignment, trimAl was introduced to provide further sequence trim.

### Hint

If both MAFFT and MUSCLE are set as default program, MAFFT is the prioritized one.

## MAFFT

### Run MAFFT

MAFFT will automatically run, do not need extra command line flags. However, users still able to run MAFFT manually by adding ‘**-mafft**’ flag.

Command-line example:

```
epita -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com -mafft
```

There are also several command line flags for changing parameters of MAFFT:

**Matrix** Command line flag of matrix selecting is ‘**-matrix [Matrix Abbreviation]**’, identical to the command line flag ‘**-bl**’ or ‘**-jtt**’ of MAFFT.

Command-line example:

```
epita -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com
```

**Opening Score And Extension Score** Command line flag of opening score setting is ‘**-op [Number]**’, identical to the command line flag ‘**-op**’ of MAFFT.

Command line flag of extension score setting is ‘**-ep [Number]**’, identical to the command line flag ‘**-ep**’ of MAFFT.

Command-line example:

```
epita -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com
```

### Tree Rebuilding Number

Command line flag of tree rebuilding number is ‘**-retree [Number]**’, identical to the command line flag ‘**-retree**’ of MAFFT. This flag can determine the guide tree built times in the progressive stage.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com
```

### Max Iterate Number

Command line flag of maximum iteration is '**-maxiterate [Number]**', identical to the command line flag '**-maxiterate**' of MAFFT. This flag can determine the cycles number of iterative refinement.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com
```

### Fast Fourier Transform Algorithm

Command line flag of choose FFTS (Fast Fourier Transform) method is '**-ffts [Mode]**', identical to the command line flag '**-localpair**', '**-genafpair**' and '**-globalpair**' of MAFFT. For each command, '**-ffts localpair**' stands for the Smith–Waterman algorithm, '**-ffts genafpair**' stands for generalized affine gap cost, '**-ffts globalpair**' stands for Needleman–Wunsch algorithm.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com
```

## MUSCLE

### Run MUSCLE

To run MUSCLE as the multiple sequence alignment program, add **-muscle** flag to the command line. MUSCLE will automatically run, do not need any parameter settings.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com -muscle
```

## trimAl

### Run trimAl

As the default setting, trimAl automatically runs in the EPTA pipeline. One can also manually add **-trimAl** flag to the command line to run trimAl.

#### trimAl Run Mode

Command line flag of selecting trimAl run mode '**-tmod [Mode]**'. There are four mode selectable, 'automated1', 'gappout', 'strict', and 'strictplus', identical to corresponding command of trimAl command line. Only 'automated1' mode is accessible in the Lite mode.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com -mafft -t
```

## Remove Spurious Sequences

The flag ‘**-rmss [Residue overlap/Sequence overlap]**’ able to remove spurious sequences when trimming multiple alignment sequences. **Residue overlap** is identical to the flag ‘**-resoverla**’ of trimAl, in charge of keep “good positions” according to a given minimum overlap of a positions with other positions in the column. **Sequence overlap** is identical to the flag ‘**-seqoverlap**’ of trimAl, which means a minimum percentage of “good positions” that a sequence must have in order to be conserved.

Command-line example:

```
epta -i ./fasta_files -o ./test -tax -name -dom -pfam -em test@example.com -mafft -ti
```

## Make a tree from alignment file

---

### Annotate a tree file

---

Under construction

## Commands

### General

---

#### -i [file or path]

---

Specify input file in normal, gunzip, or tar.gz, or a path includes all needed input files.

#### -o [file or path]

---

Specify output path.

#### -em [email address]

---

Email address that you want receive potential information from web tools.

### Parse FASTA

---

#### -dh

---

Keep duplicate headers.

#### -tax

---

Search for taxonomy information for each sequence in Entrenz database.

#### -name

---

Search for protein name for each sequence in Entrenz database.

#### -dom

---

Annotate protein domians on the tree.

## –pfam

---


Run Pfamsacn (domain prediction) for each sequence.

## –pev [E-value]


---

E-value of Pfamscan search.

Available E-value:



50  
20  
10  
5  
2  
1  
0.1  
0.01  
0.001  
0.0001  
1e-5  
1e-10  
1e-50  
1e-100  
1e-300



## –pas

---

Enable active sites prediction in Pfamscan.

## Make alignment

---

Under construction

## Make phylogenetic tree

---

Under construction

## Tree Annotating

---

Under construction