FlaGs analyzes genomic context around genes encoding protein of interest using python script combined with *Jackhmmer* and *ETE-toolkit* (optional). This tool aims to find the flanking genes of the genes of interest, cluster them based on the homology and represent them visually with distinguished identifiers (color and number).

At initiation the tool verifies the input file which is a list of protein accession. Since one protein could be present in multiple species, FlaGs takes each protein accession and finds the corresponding list of Genome Assembly Number (eg. *GCF\_000001635*.*23 or GCA\_000001635.5)* which is specific for each species. It uses NCBI RefSeq and GenBank databases to verify the accession number. User can restrict the verification and searching of flanking genes for the protein accession using a tab delimited input file containing the list of Genome Assembly number and the protein accession number.

After verification of query list, FlaGs uses the Genome Assembly number to retrieve the Genomic Feature Format (gff3) file for each protein query and use the information to identify annotated flanking genes from upstream and downstream regions. Then it retrieves the sequence for all flanking genes and search for homology using *Jackhmmer* tools. Then it clusters the flanking genes based on homology and each cluster is assigned by a specific number. The numbering of cluster begins with 1. The lower the cluster number the more frequent it is. Finally, it visualizes the output using tkinter module. Black pentagonal structure represents the query proteins and rest represents the flanking genes. The structure represents strand information along with length of the gene and the gap between the structure represents the intergenic space. The number and color represent the cluster.

ETE-toolkit provides an additional feature to FlaGs. It generates a phylogenetic tree using the query sequences. It uses “mafft\_default-trimal01-none-fasttree\_full” workflow to generate tree. The flanking genes and the queries are represented as flags (Triangular) like structure.

**Prerequisite**

1. Python

I used Anaconda Python v.3.5.

2. Biopython

To install Biopython I used following command:

conda install -c bioconda biopython

3. Jackhmmer

conda install -c biocore hmmer

User can download the hmmer programme from <http://hmmer.org/> and install according to the instruction.

4. ETE-toolkit

conda install -c etetoolkit ete3

**Usage :**

./FlaGs.py -h

**Arguments:**

1. "-a" "--assemblyList"

Protein Accession with Genome Assembly Identifier eg. GCF\_000001765.3 in a text input file separated by tab.

Input File Example:

GCF\_000001765.3 WP\_047256880.1 #tab separated

GCF\_000002753.1 WP\_012725678.1

…….

2. "-p", "--proteinList"

Protein Accession eg. XP\_ or WP\_047256880.1 in a text Input file separated by newline.

Input File Example:

WP\_047256880.1

WP\_012725678.1

…….

3. "-e", "--ethreshold"

This e-value is used as a cutoff parameter to detect homology among all flanking genes. By default it is 1e-10.

4. "-n", "--number"

This number represents number of Jackhmmer iterations that allows to find more distant homolog, by default it is set as 3.

5. "-s", "--stream"

Using this parameter in FlaGs user can define number of upstream and downstream genes to look for each query in input list. By default, it is 4, which means for each query it will try to find 4 upstream and 4 downstream flanking genes and then process further.

6. "-t", "--tree"

It requires ETE3 installation. This option enables to show flanking genes along with phylogenetic tree, but for that user needs to use this parameter.

7. "-ts", "--tshape"

It requires ETE3 installation and thus this option only works when -t is used. This parameter can increase or decrease the size of triangle shapes that represent flanking genes, by default it is 12.

8. "-tf", "--tfontsize"

It also requires ETE3 installation and thus this option only works when -t is used. This parameter can increase or decrease the size of font inside triangles that represent flanking genes, by default it is 4.

9. "-o", "--out\_prefix"

Any Keyword to define your output eg. MyQuery.

10. "-k", "--keep"

If user wants to keep the intermediate files eg. gff3, this option is useful then. By default, it will remove.

11. "-v", "--version"

Version number of the program.

12. "-vb", "--verbose"

User can use this option to see the work progress for each query as STDOUT.