clusterTools User Manual

Emmanuel LC de los Santos (e.de-los-santos@warwick.ac.uk)

https://www.github.com/emzodls/clustertools
July 5, 2016

Contents

1 Introduction

clusterTools is a set of Python libraries for help in analysis of biosynthetic gene clusters. It contains different methods for communicating with sequence databases, parsing output from different programs, and organizing the results of these queries in a format that makes it convenient to make systematic queries. We have used this toolkit to search the public domain for putative polyketide alkaloid gene clusters

1.1 Installation and Requirements

clusterTools uses the following Python packages:

- Biopython (`http://biopython.org`)
- Numpy (`http://www.numpy.org/`)
- Scipy (`https://github.com/scipy/scipy/releases`)
- bx-python (`https://github.com/bxlab/bx-python`)
- Sorted Containers (`http://www.grantjenks.com/docs/sortedcontainers/`)
- reportlab (`https://bitbucket.org/rptlab/reportlab`)

For MacOS, you can install these packages easily via homebrew and pip.

1.2 Components

clusterTools consists of four separate modules. These are:

- clusterGenbank - interface to genbank database, methods for parsing genbank files to clusterTools compatible databases.
- clusterAnalysis - core analysis module. contains Protein and Cluster objects.
- clusterDistance - functions to compare two different cluster objects.
- clusterGraphics - functions for automatically visualizing Cluster objects

2 clusterTools Modules

2.1 clusterAnalysis

clusterAnalysis is the main analysis module. It defines the Protein and Cluster objects which are used by the other modules. It contains functions to parse clusterTools compatible fasta files, and output from BLAST and hmmer to integrate the information, allowing the user to make different queries based on different hypotheses (see included sample file that runs through a workflow for finding putative polyketide alkaloid clusters for usage).

2.1.1 Objects

- Protein

  Container for different information about proteins. At minimum, this contains the genbank accession number for both the nucleotide source (species) and the protein coding sequence (name), its physical nucleotide position in its source (location). Additional information from homology searches, hmmer queries, and its amino acid sequence are optional. Protein objects can be defined by parsing a clusterTools compatible fasta file, or parsing hmmer or BLAST queries against clusterTools compatible fasta databases.

  Functions of this object include:

  - calculate_distance(protein, hit_dict_id) - calculates the distance between two protein objects based on a BLAST query. The BLAST query is expected to take the single highest scoring alignment between two proteins. If no such alignment is stored in the protein objects, this distance is 1.
  - getDomStr(annotID,delimiter) - if hmmer output is included this will return a sequence of hmmer domain hits the protein is annotated with.
  - getAnnotations(annotID) - returns a set with the different hmmer hits for a given hmmer query.

- Cluster

  The Cluster object is an extension of a SortedList of Protein objects. The list is ordered based of the physical location of each of the proteins in relation to one another from 5' to 3' on the DNA.

2.1.2 Functions

- add_sequences(fastaFile,proteinDict)

  Given a clusterTools compatible fastafile and a proteinDictionary (key=reference to protein object,item = protein object). Returns the proteinDictionary with additional proteins from the fastaFile included. Creating Protein objects in this way means that the sequence information is embedded in the protein object.

- parse_allVall_blast_file(path, proteinDict, swapQuery=False, evalCutoff=10, scoreCutoff=0)

  Parse blast tabular output and add the protein sequences to the protein dictionary. You have the option to create protein objects based on the subject or query sequences, this affects what sequences are protein objects and what the hits of the protein objects are. You can alter the eval and score that the parser will consider to be significant. Returns the proteinDictionary with the BLAST information added.

  Recommended BLAST settings are:

```
makeblastdb -dbtype prot -in allOutput_CDS_prot.fasta

blastp -db allOutput_CDS_prot.fasta -query allOutput_
```

- parse_hmmsearch_domtbl_anot(path, minDomSize, anotID, proteinDict, eval_cutoff=1e-5, cutoff_score=25, verbose = False, speciesFilter = None, proteinFilter = None)

  Parse hmmsearch tabular output to annotate a protein with different domains. Annotations will show up in the protein under the specified *anotID*. You can set the minimum evalue and the score that the parser will consider to create or augment the protein objects in the protein dictionary. Will return the input proteinDictionary with the annotations from the hmmsearch output. *speciesFilter* and *proteinFilter* are optional sets that can be given to speed up the parser in the case of large hmmsearch outputs. *speciesFilter* will only consider entries that have the species accession numbers in the set. *proteinFilter* will only add annotations to the proteins in the set.

- merge_annotations(protein, mergedID, anot1, anot2, minDomSize, delOrig = False)

  Will merge two different annotations from separate hmmsearch queries into a single input based on a set of rules (see comments in resolve_conflicts for more details). This allows you to combine information from multiple hmmsearch queries for your analysis.

2.2 clusterGenbank

clusterGenbank is composed of functions to connect to the NCBI genbank database to fetch sequences and to format genbank files for use with other clusterTools modules.

2.2.1 Functions

- batch_process(genbank_file_list, outputPath = '.',speciesOverride = None, offSet = 0, upstream_size = 200, inclNtCDS = True, inclProm = True,singleFile=False,includeORF = False)

  This is the main method of clusterGenbank. Given a list of paths to genbank files (can be generated by the *glob* module), this function will process the genbank files into a single (with *singleFile*) or multiple fasta files that the amino acid sequences and physical location in the DNA of all the CDS sequences in the genbank file that external programs such as BLAST or hmmer can use. Fasta entries are formatted in a way that is compatible wiith the clusterAnalysis parsing functions in order to make easy queries. Options include:

  - speciesOverride - a string that will be used for naming the coding sequences

- offSet - for use when parsing genbanks from antiSMASH's clusters since coding their nucleotide sequences all start at 1, the offSet adjusts this to correspond to their actual location in the DNA.
- upstream_size - only useful if *inclProm* is True. This is the amount of nucleotide sequence upstream of the coding sequence that will be considered as the promoter.
- inclNtCDS - will generate a separate fasta file that contains the nucleotide sequence of the CDS entries along with the amino acid sequences.
- outputPath - path to where fasta files will be written
- inclProm - will generate a separate fasta file containing the *upstream_size* nucleotides upstream of each of the CDS entries
- singleFile - will only generate a single file of amino acid sequences from the list of genbank files
- includeORF - aside from sequences annotated as CDS entries will also include ORF feartures

• fetchClusterGenbanks(clusterDict,window,targetDir ="./", writeSummary =False)

  fetchClusterGenbanks takes a clusterDict object (generated by clusterAnalysis) and fetches the nucleotide sequences associated with the query within a certain *window*, which is the total length of the nucleotide sequence it will fetch. The nucleotide sequence is centered around the midpoint of the genes specified in the clusterDict object. These genbank files can be used for analysis with an external program (such as antiSMASH)

• fetchGbNucl(targets,window,targetDir ="./",writeSummary = False)

  Similar to fetchClusterGenbanks, but takes a list of tuples containing the genbank accession number and the location of the putative cluster as input.

• generateSpeciesDict(accList)

  Given a list of genbank accession numbers, this will return a dictionary with the accession numbers as a key, and the name of the entry corresponding to the accession number as the item.

2.3 clusterDistance

This contains two functions to compare cluster objects based on a specific homology search. The cluster distances are calculated slightly differently, but the core algorithm is the same: First, the pairwise distances of each of the proteins in the cluster is computed using the calculate_distance function of the protein object. Second, the proteins are paired up to minimize the sum total distance between the clusters

using the hungarian algorithm. Finally, the cluster distance score is calculated and normalized, resulting in a value between 0 and 1.

2.3.1 Functions

- calculateClusterDist(cluster1,cluster2,hitDictID,linearDist = False)

  Calculates distance by adding the distances between protein pairs normalized to the proportion of each protein with respect to the cluster. Will return the calculated distance and a list of tuples corresponding to the proteins that were paired by the hungarian algorithm.

- calculateDistBitScore(cluster1,cluster2,hitDictID)

  Extension of the way distances are calculated for proteins. Subtracts from 1 the cumulative bitscore of all of the pairs minus the maximum bitscore of the cluster. This is normalized by the ratio of the clusterSizes between the two clusters. Will return the calculated distance and a list of tuples corresponding to the proteins that were paired by the hungarian algorithm.

2.4 clusterGraphics

2.4.1 Functions

- generateClusterCompGraphic(cluster1,cluster2,pairs,outname)

  Given two cluster objects and the pairing between the proteins in the cluster (generated by either function of clusterDistance), will write an svg file comparing the two given clusters. This will contain the protein layout of each of the clusters and a graphic showing which proteins in the cluster are paired.