CGPFinder

Preprocessing steps

In order to run CGPFinder, a proteome containing all the proteins for non-overlapping genes in the species under consideration is needed. In addition, a csv table with annotation for such genes is required.
This document describes the steps that were taken to obtain and parse the above mentioned requirements.

The scripts mentioned in this document are stored in the directory extras.

1. **Download proteomes and GenBank files**:
   - Input: `species.txt`. A file containing the names of the species under study, with one species name per line, and spaces converted to underscores. Important: If a species does not have a "reasonably assembled genome" (existing "Assembled_chromosomes" directory in its NCBI ftp), the script will fail.
   - Process: `get_genomes.py`: By running this script with no arguments, it will download the proteomes for each species and the GenBank files for each chromosme of the reference assembly, excluding unplaced and unlocalized "chromosomes".
2. **Manual step a**: This step was not automated in order to make sure that the correct files were included. Here, non-relevant chromosomes should be removed. Sometimes, particular genome annotation projects can change the names given to unassembled chromosomes, and these should be removed.
3. **Manual step b**: When careful manual inspection of the chromsome files has been done, all the GenBank files should be concatenated to a file named `genomes.gb`. Similarly, all the proteomes (Species_name.fa) should be concatenated to a file named `proteomes.fa`
4. **Parsing genomes**: In this step, `genomes.gb` is going to be parsed, overlapping genes removed, and proteome sequence names standardized. The script for this step is `preprocessing.py`. It is set to be run in the same directory where the files `genomes.gb` and `proteome.fa` are. The output files are described below:
   - `genes_cds.csv`: First pass of the parsing process. Annotation of all CDS features in `genomes.gb`
   - `mini_table.csv`: Subset of columns from `genes_cds.csv`
   - `genes_parsed.csv`*: Result of parsing process. It contains annotation for non-overlapping genes
   - `chromosomes.csv`*: Chromosome annotation. Lengh, assembly GI, etc.
   - `all_seqs.fa`*: Sequence file containing the protein sequences of the genes in `genes_parsed.csv`. In order to make post-processing easy (extracting interesting sequences, phylogenetics, etc). The sequence name of the protein sequences in `all_seqs.fa` follows the following format: species|chromosome|accession|symbol|start|end|strand

   *: File needed by CGPFinder. It should be placed in the directory to be used as database with the --db parameter in cgp_exe.py.
5. **Blast database**: In order to speed-up the sampling process. A blast all vs. all of all the proteomes (in `all_seqs.fa`) should be run, and the results stored in json files (one per species). This is accomplished in two steps:
   1. **Run blast**: `all_blasts.py` will generate species-specific fasta files. The script should be run in the directory `blasts` under the directory containing the blast database and annotation files that is going to be used in the --db parameter in `cgp_exe.py`. Then, a blast search of each species' sequences against `all_seqs.fa` will be executed. Importantly, the same parameters, filters and parsing algorithms that CGPFinder uses will be used in this step. The output is described below. `all_blasts.py` should be run as follows: `python all_blasts.py ../all_seqs.fa <num_cores>`.The output is described below:
      - `Species_name.fasta`: Species-specific proteome
      - `Species_name.blast`: Raw blast output in table format
      - `Species_name.blast_filtered`: Hits where the subject was more than 1.5 times longer or less than 0.3 shorter than the query are removed
      - `Sepecies_name.blast_out`: Final parsed table. This file parses `Species_name.blast_filtered` to conform with the annotation format needed to generate the json files
6. **Generate JSON files**: With `hits_dictionary.py`, the `Species_name.blast_out` files will be parsed to json files. This will make their loading in the sampling step in CGPFinder more efficient. `hits_dictionary.py` recieves the directory where the *.blast_out files are stored and the location of `genes_parsed.csv`. An example run will look like `python hits_dictionary blast_dir path/to/genes_parsed.csv`. json files will be generated in the directory set by the first argument (blast_dir, in the example). CGPFinder expects the json files to be located under the directory `blasts`, which should be in the same directory as the directory to be used as database by the –db argument in cgp_exe.py.