pyrewton

Release 0.1.1

Aug 22, 2023

Contents

1 Overview **3**

2 Contents **5**

2.1 The genbank module . . . 5
2.1.1 get_ncbi_genomes . . . 5
2.1.2 get_genbank_annotations . . . 6
2.2 The cazymes module . . . 7
2.2.1 uniprot . . . 7
2.2.2 prediction . . . 9
2.3 Trouble Shooting . . . 10
2.3.1 genbank: get_nbci_genomes . . . 10
2.3.2 get_genbank_annotations . . . 11
2.4 Notebooks . . . 11
2.4.1 Accessing the notebooks . . . 11
2.5 License . . . 12
2.5.1 MIT License . . . 12
2.6 Questions? . . . 12

3 Requirements **13**

4 Installation **15**

5 Notebooks **17**

6 Help, Contribute and Support **19**

Version v0.1.1 2020/06/04
DOI: 10.5281/zendo.3876218
GitHub repository

CHAPTER 1

Overview

Pyrewton is a Python3 package for the identification of Carbohydrate Active enZymes (CAZymes) from candidate species, providing the user a complete CAZyome (all CAZymes encoded within a genome) for each candidate species. Pyrewton invokes and statistically evaluates three CAZyme prediction tools [dbCAN](https://github.com/linnabrown/run_dbcan), [CUPP](https://www.bioengineering.dtu.dk/english/researchny/research-sections/section-for-protein-chemistry-and-enzyme-technology/enzyme-technology/cupp), and [eCAMI](https://github.com/zhanglabNKU/eCAMI).

Pyrewton is designed to be run at the command line and is free to use under the MIT license, with proper recognition.

Pyrewton supports: - Downloading of all genomic assemblies (as GenBank files .gbff) from the [NCBI Assembly database](https://www.ncbi.nlm.nih.gov/assembly) associated with each candidate species passed to the programme - Retrieval of all annotated protein sequences from GenBank (.gbff) files - Retrieve proteins entries from [UniProtKB](https://www.uniprot.org/), using a JSON file to configure the queries. Writing out the protein data to a summary dataframe and protein sequence to a FASTA file.

Features currently in development: - Use the 3rd-party tools dbCAN, CUPP and eCAMI to predict which proteins within a FASTA file (generated by searching genomic assemblies and querying UniProt) are CAZymes - Evaluate the accuracy of the CAZyme prediction tools to distinguish between CAZyme and non-CAZyme protein sequences - Evaluate the accuracy of the CAZyme prediction tools to the correct CAZy family - Produce a report of the CAZyme prediction tool evaluation

More detailed documentation for each module is linked to in the contents table below, including links to documentation to help with trouble shooting.

CHAPTER 2

Contents

2.1 The genbank module

The module `genbank` contains submodules that handle GenBank files. This includes the retrieval of GenBank files from the NCBI Assembly database, and retrieval of protein annotations (writing out th protein sequences to FASTA files and protein data to a summary dataframe) from GenBank files.

- *get_ncbi_genomes*
- *get_genbank_annotations*

2.1.1 get_ncbi_genomes

`get_ncbi_genomes` is a submodule of the `genbank`, it takes a plain text file containing species scientific names or NCBI taxonomy as input. The script finds the corresponding taxonomy ID or scientific name, as appropriate, as well as retrieve all directly linked accession numbers. The submodule generates a dataframe containing 'Genus', 'Species', 'NCBI Taxonomy ID', and 'NCBI Accession Number'.

Invoking the script from the command line

To invoke `get_ncbi_genomes` invoke the script `get_ncbi_genomes.py`, for example, using the following command: `python3 get_ncbi_genomes <user_email> <-i path_to_input_.txt> <-o directory_to_store_assemblies> <-d species_dataframe_output_path`

Compulsary arguments

`-u, --user` - Although indicated as optional, Entrez requires an email address must be provided. If not provided the programme will log this as an error and terminate.

Optional arguments

`-d, --`dataframe - Specify output path for dataframe, which will be saved as a .csv file (inclusion of file extensions is optional). If not provided dataframe will be written out to STDOUT.

`-f, --force` - Enable writting in specificed output directory if output directory already exists.

`-g, --genbank` - Enable or disable downloading of GenBank files.

`-h, --help` - Display help messages and exit, including listing arguments

`-i, --input` - Specify input filename (with extension) input file.If only the filename is supplied `get_ncbi_genomes.py` will only look in the curent working directory, otherwise provide path to input file. If no option is given the default input is taken from STDIN.

`-l, --log` - Specify name of log file (With extension). If only filename is given, log file will be written out to the current working directory, otherwise provide path including filename. If not option is given no log file will be written out, however, logs will still be printed to the terminal.

`-n, --nodelete` - Enable not deleting files in existing output directory. If not enabled, output directory exists and writing in output directory is 'forced' then files in output directory will not be deleted, and new files will be written to the output directory.

`-o, --output` - Specify filename (with extension) of output file. If not option is given output will be written to STDOUT.

`-r, --retries` - Specifiy maximum number of retries before cancelling call to NCBI if a network error is encountered. The default is a maximum of 10 retries. When maximum is reached, a value of 'NA' is returned.

`-t, --timeout` - Specify timeout limit of URL connection when downloading GenBank files. Default is 10 seconds.

`-v, --verbose` - Enable verbose logging - changes logger level from WARNING to INFO.

2.1.2 get_genbank_annotations

`get_genbank_annotations` is a submodule of the `genbank`, which reads NCBI GenBank files and retrieves annotations of encoded Proteins. Specifically, `get_genbank_annotations` parses each genomic assembly contained within the input directory passed the module, during which it identifies a protein by its annotation as a 'CDS' feature. The summary data (listed out below) for every protein is written out to a dataframe, with a unique protein in each row. The sequence of each protein is also written out to a FASTA file. Proteins from the same genomic assembly are written out to the same FASTA file, therefore, a singel FASTA file containing the sequences of all the proteins identified within the assembly is created per genomic assembly in the input directory.

Dataframe columns: * 'Genus' * 'Species' * 'NCBI Taxonomy ID' * 'NCBI Accession Number' * 'NCBI Protein ID' * 'Locus Tag' * 'Gene Locus' * 'NCBI Recorded Function' * 'Protein Sequence' * 'UniProtKB Entry ID' * 'UniProt Entry Name' * 'UniProtKB Protein Names' * 'EC number' * 'Length (Aa)' * 'Mass (Da)' * 'Domains' * 'Domain count' * 'UniProtKB Linked Protein Families' * 'Gene ontology IDs' * 'Gene ontology (molecular function)' * 'Gene ontology (biological process)'

`get_genbank_annotations` has been created as a subsequent or complementary module to `get_ncbi_genomes` and is designed to parse the outputs from `get_ncbi_genomes`.

Invoking the script from the command line

To invoke `get_genbank_annotations` invoke the script `get_genbank_annotations.py`, for example, use the following command: `python3 get_genbank_annotations <path_to_input_df> <path_to_assemly_dir>`

Compulsary arguments

`--input_df` - Path to the dataframe containing the accession numbers of the genomic assemblies. This dataframe is created by `get_ncbi_genomes`.

`--genbank` - Path to directory containing the genomic assemblies listed within the dataframe. Unless they have been moved, this is the output directory from `get_ncbi_genomes`.

Optional arguments

`-h, --help` - Display help messages and exit, including listing arguments

`-f, --force` - Enable writting in specificed output directory if output directory already exists.

`-l, --log` - Specify name of log file (With extension). If only filename is given, log file will be written out to the current working directory, otherwise provide path including filename. If not option is given no log file will be written out, however, logs will still be printed to the terminal.

`-n, --nodelete` - Enable not deleting files in existing output directory. If not enabled, output directory exists and writing in output directory is 'forced' then files in output directory will not be deleted, and new files will be written to the output directory.

`-o, --output` - Specify filename (with extension) of output file. If not option is given output will be written to STDOUT.

`-v, --verbose` - Enable verbose logging - changes logger level from WARNING to INFO.

Note: get_genbank_annotations is still under development. Please see the GitHub repository for the latest developments.

2.2 The cazymes module

The module `cazymes` contains submodules that automate the retrieval of protein sequences and associated annotations from UniProtKB, invokes the CAZyme prediction tools dbCAN, CUPP and eCAMI to predict which proteins within a FASTA file are CAZymes, and evaluates the performance of the CAZymes prediction tools.

- *uniprot*
- *get_genbank_annotations*

2.2.1 uniprot

The `uniprot` submodule automates the querying and parsing of data from [UniProtKB](https://www.uniprot.org/). The querying of UniProtKB is configuerable via a yaml file. For every protein retrieved from UniProtKB the protein sequence is written to a FASTA file (with a single FASTA file per query to UniProtKB), and the following data for each protein is written out to a summary dataframe (writing a dataframe per UniProt query):

- 'NCBI Taxonomy ID'
- 'Organism'
- 'UniProtKB Entry ID'
- 'UniProtKB Entry Name'
- 'UniProtKB Protein Names'
- 'EC number'

- 'Length (Aa)'
- 'Mass (Da)'
- 'Domains'
- 'Domain count'
- 'UniProtKB Linked Protein Families'
- 'Gene ontology IDs'
- 'Gene ontology (molecular function)'
- 'Gene ontology (biological process)'
- 'Sequence'

Configuring the querying of UniProt

An example configuration yaml file is located within the directory pyrewton/cazymes/uniprot, called 'uniprot_cazy_ec_retrieval.yaml'. Specific queries can be written under the 'queries' tag in a yaml file. These queries must be written in the [UniProt syntax](https://www.uniprot.org/help/text-search). A list of UniProt query fields is available [here](https://www.uniprot.org/help/query-fields).

To restrict these queries to a specific set of candidate species add the NCBI taxonomy ID of the species under the 'tax_id' in the yaml file. Specifically, this causes every query that is listed under the 'queries' tag to be performed for each species listed under the 'tax_id' tag, saving time having to write out the same set of queries multiple times over with a different taxonomy ID. If no taxonomy ID is given under the 'tax_id' tag, and no taxonomy ID is included within the query under the 'queries' tag the search will no be restricted to a specific species.

Providing specific queries under the 'queries' tag of the configuration yaml file is also optional. If only the taxonomy ID of a species is given and there are no queries under the 'queries' tag, all proteins for that given species will be retrieved from UniProtKB.

If you are looking to retrieve all potential CAZymes from UniProtKB for a set of candidate species the example yaml configuration file 'uniprot_cazy_ec_retrieval.yaml' contains EC numbers retrieved from the [CAZy database](http://www.cazy.org/) and are strongly associated with CAZyme activity, and also contains a query to retrieve all proteins that are catalogued in both UniProtKB and CAZy. Therefore, it is suggested this file is used as the configuration file by either changing the taxonomy id to the your candidate species NCBI taxonomy IDs or removing all taxonomy IDs and retrieving all proteins which are linked to CAZy or annotated with one of the specified EC numbers from UniProtKB.

Invoking the script from the command line

An example of basic operation is: `python3 get_uniprot_proteins <path_to_config_file.yaml>`

Compulsary arguments

`-u, --user` - Although indicated as optional, Entrez requires an email address must be provided. If not provided the programme will log this as an error and terminate.

Optional arguments

`-a, --fasta` - Enable writing out FASTA files containing protein sequences

`-f, --force` - Enable writting in specificed output directory if output directory already exists.

`-h, --help` - Display help messages and exit, including listing arguments

`-l, --log` - Specify name of log file (With extension). If only filename is given, log file will be written out to the current working directory, otherwise provide path including filename. If not option is given no log file will be written out, however, logs will still be printed to the terminal.

`-n, --nodelete` - Enable not deleting files in existing output directory. If not enabled, output directory exists and writing in output directory is 'forced' then files in output directory will not be deleted, and new files will be written to the output directory.

`-o, --output` - Specify output directory where all output dataframes and FASTA files are written to If not option is given output will be written to STDOUT.

`-v, --verbose` - Enable verbose logging - changes logger level from WARNING to INFO.

2.2.2 prediction

The `prediction` module is for the prediction of CAZymes and non-CAZymes from protein sequences within FASTA files. The module is setup to take the output from from the `uniprot` and `get_genbank_annotations` submodules, although it will accept any FASTA file containing protein sequences.

This module invokves three third-party CAZyme prediction tools that predict if a given protein sequence is a CAZyme or non-CAZyme, and then predicts the CAZy family of any predicted CAZymes. These tools include [dbCAN](https://github.com/linnabrown/run_dbcan), [CUPP](https://www.bioengineering.dtu.dk/english/researchny/research-sections/section-for-protein-chemistry-and-enzyme-technology/enzyme-technology/cupp), and [eCAMI](https://github.com/zhanglabNKU/eCAMI).

This module also evaluates the overall performance of each of the prediction tools, but also evaluates the performance of each prediction tool per input FASTA file. The latter allows the evaluation of each prediction tool per candidate species if a each input FASTA file contains proteins from a single species/genomic assembly. In order to perform this evaluation a gold standard is required. CAZy is the gold standard database for all identified and catalogued CAZymes, but does not offer a method of automated protein data retrieval, therefore, use this [CAZy webscrapper](https://github.com/HobnobMancer/cazy_webscraper) to automate the retrieval of the necessary protein data from CAZy. Using this web scraper will ensure the data retrieved from CAZy is in the correct format for the statistical evaluation of the performnce of the dbCAN, CUPP and eCAMI.

The output

For every input FASTA file a respective output directory is created within the user specified output directory. In this way the output directory specified by the user becomes the parent directory of all output directories from the module. Within each input FASTA files output directory there will be:

- The output directory from dbCAN (becuase dbCAN creates its own output directory)
- A fasta and log output files from CUPP
- An output .txt file from eCAMI
- A standardised output dataframe for each prediction tool (therefore, one output dataframe for dbCAN, DIAMOND, HMMER, Hotpep (which are invokved within dbCAN), CUPP and eCAMI)
- Results of the statistical evaluation of prediction tools performance
- Report presenting the evaluation of the prediction tools performance

The parent output directory (the output directory specified by the user) will also contain the results and report of the statistical evaluation of the prediction tools' overall performance across all input FASTA files passed to the module when the module was invoked.

Invoking the script from the command line

This script is still under development.

Note: get_genbank_annotations is still under development. Please see the GitHub repository for the latest developments.

2.3 Trouble Shooting

This section covers common errors expected to arise when invoking each module/submodule, and the probable causes. If any other issues arise, please raise any issues with any of the programmers at the GitHub repository issues pages, by following the link.

- genbank-mod-get-ncbi-genomes
- *get_genbank_annotations*

2.3.1 genbank: get_nbci_genomes

This section deals with troubleshooting the genbank module's submodule get_ncbi_genomes, for the retrieval of GenBank files from the NCBI Assembly database.

Warning: The majority of issues will arise due to errors in the input file. Always ensure the input file does not contain any error or blank lines. Additionally, always ensure the correct path to the input file is provided.

IOError This error will occur if there is a network issue when using Entrez to call to NCBI. The script will automatically retry the call the set maximum number of times. If the maximum number of retries is met before connecting to NCBI without encountering a network error, 'NA' is returned and stored in the dataframe.

FileNotFoundError This error will occur is the incorrect path is provided as the input argument at the command line, or no input argument is provided and STDIN contains no data. Ensure the path includes the file name, with extension. If this error occurs the program will terminate.

IndexError during scientific name retrieval This occurs when Entrez fails to retrieve a scientific name for the given taxonomy ID. This is potentially caused by a typo in the taxonomy id provided in the input file. If this error occurs the string 'NA' will be returned.

IndexError during taxonomy ID retrieval This occurs when Entrez fails to retrieve a taxonomy ID for the given scientific name. Returns 'NA'. This is potentially caused by a typo in the species name in the input file, or a typo in a taxonomy ID 'NCBI:txid' prefix, causing the program to misinterpret the ID as a species name and use it to try and retrieve a scientific name. If this error occurs the string 'NA' will be returned. If no taxonomy ID is available for the retrieval of accession numbers, the retrieval of accession numbers is cancelled, and a value of 'NA' is returned.

IndexError during assembly ID retrieval This occurs when Entrez fails to retrieve assembly IDs from NCBI. This may be because there are no directly linked assemblies for the given taxonomy ID. Check the NCBI Taxonomy database to ensure there are 'directly' linked assemblies and not only 'subtree' assemblies. If this error occurs the program with exit the retrieve of the assembly IDs and not retrieve the NCBI accession numbers, and return the string 'NA'. This allows for troubleshooting using on the specie(s) for which it is required, to reduce demand on NCBI.

RuntimeError during posting of assembly IDs to NCBI This error occurs when Entrez fails to post the retrieved assembly IDs, causing it to fail to retrieve the document summary. This is potentially caused by incorrect formatting of the assembly IDs or the request is too large for Entrez/ NCBI. If this is the case, repeat the procedure in batches. If this error occurs the program with exit the posting of the assembly IDs and not retrieve the NCBI accession numbers,

and return the string 'NA'. This allows for troubleshooting using on the specie(s) for which it is required, to reduce demand on NCBI.

RuntimeError or IndexError during retrieval of accession numbers This error occurs when Entrez fails to retrieve the document summary from NCBI. This is potentially caused by incorrect formatting of the assembly IDs or the request is too large for Entrez/ NCBI. If this is the case, repeat the procedure in batches. If this error occurs the program with exit the retrieval of the NCBI accession numbers, and return the string 'NA'. This allows for troubleshooting using on the specie(s) for which it is required, to reduce demand on NCBI.

2.3.2 get_genbank_annotations

This section deals with troubleshooting the genbank module's submodule get_cazyme_annotations, for retrieving cazyme annotations from GenBank files.

2.4 Notebooks

Jupyter notebook environments have been used to facilitate explanation of the function and operation of pyrewton. These notebooks explicitly describe how the scripts were invoked to during the EastBIO PhD project ('Identificating Engineering Candidate for Advanced Biocatalysis in Biofuel Production'), and were written to supplement the thesis. However, these notebooks can also be used as an example as to invoking `pyrewton` to identify cazyme canidated for engineering. The notebooks all containing the final output generated when invoked during the EastBIO PhD project.

Note: These notebooks contain exerts of code to facilitate the explanation of the code code architecture and function, as well as details provided on how the code was implemented during the project. Therefore, the code in many of the code cells is not runnable and replication or exploration of the data should be performed using the original scripts provided within the repository.

2.4.1 Accessing the notebooks

The notebooks can be accessed via the GitHub pages created for the host repository, or directly via the list below.

Accessing the notebooks in browser

01_downloading_genbank_files This refers to invoking get_ncbi_genomes to download all directly linked GenBank files for each species passed to the program.

02_retrieving_genbank_annotations This refers to invoking get_genbank_annotations to retrieve all proteins annotations from the downloaded GenBank files.

Accessing the notebooks via the terminal

From the top level of the repository directory on your system, start the Jupyter notebook server by issuing the command: `Jupyter notebook`. This will open a new browser window or tab containing the Jupyter homepage, with a listing of all files and directories as laid out in the repository. Navigate to the 'notebooks/' directory, containing all Jupyter notebooks for the repository. To open a notebook simple click on the title of the notebook. For more information please see the Jupyter notebook Quick-start guide and this tutorial.

2.5 License

2.5.1 MIT License

2.6 Questions?

Please raise an issue at the GitHub repository issues pages, by following the link.

CHAPTER 3

Requirements

Python version 3.7+ Miniconda3 managed microenvironment, incorporated code checkers are included in list form in ‘requirements.txt’. Miniconda3 environment file is also available in the GitHub repository: ‘environment.yml’.

CHAPTER 4

Installation

1. Navigate the directory you wish to store pyrewton in, then clone this repository. *git clone https://github.com/HobnobMancer/pyrewton.git*

1. Create a virtual environment with dependencies, then activate the environment. *conda create -n <venv_name> python=3.8 diamond hmmer prodigal -c conda-forge -c bioconda conda activate <venv_name>*

2. Install all requirements from requirements.txt file. The requirements.txt file is stored in the root of this repository. *pip3 install -r <path to requirements.txt file>*

3. Install pyrewton. *pip3 install -e <path to directory containing setup.py file>* Do not forget to use the **-e** option when install using pip3, otherwise each time pyrewton is invoked a ModuleNotFound error will be raised. Pass the path to the **directory** containign the setup.py file not the path to the setup.py file; if you are currently in the root directory of the repoistory where the file is located, simply use '.' to indicate the current working directory.

4. Install third party CAZyme predicition tools.

To install dbCAN follow the instructions within their [GitHub repository](https://github.com/linnabrown/run_dbcan), **BUT ignore** steps 1 and 2 of their installtion guide, becuase the necessary virtual environment was already created in the second step of this installation and it meets all requirements of dbCAN. Install dbCAN within **'pyrewton/cazymes/prediction/tools/dbcan'** directory within the repository, otherwise pyrewton will not be able to find the tool.

To install eCAMI follow the instructions within their [GitHub respository](https://github.com/yinlabniu/eCAMI). eCAMI must be installed within the directory pyrewton/cazymes/prediction/tools/ecami. Following the method from the eCAMI repository will write eCAMI to 'pyrewton/cazymes/prediction/tools/ecami/**eCAMI**', to avoid this perform the installation within 'pyrewton/cazymes/prediction/tools' and rename 'eCAMI' to 'ecami', thus install eCAMI in **'pyrewton/cazymes/prediction/tools/ecami'**.

To install CUPP download the CUPP files from the [DTU Bioengineering server](https://www.bioengineering.dtu.dk/english/ResearchNy/Research-Sections/Section-for-Protein-Chemistry-and-Enzyme-Technology/Enzyme-Technology/CUPP), and store the files in **'pyrewton/cazymes/prediction/tools/cupp'**. It is not necessary to download all the files becuase the .tar and .tar.gz directories each contain all the files, therefore, download either the .tar _or_ .tar.gz directories and unpackage them or download all the files located within 'CUPP_v1.0.14'.

CHAPTER 5

Notebooks

Jupyter notebook environments were crated, documenting how pyrewton was used during the EastBIO 2019-2023 PhD Project, the GitHub pages for which are available here. These can be used as examples for how to use pyrewton in research.

CHAPTER 6

Help, Contribute and Support

Many of the common errors expected to arise during the operation of the scripts provided in this repository are covered in this documentation, including the probable causes of these issues.

Please raise any issues with any of the programmers at the GitHub repository issues pages, by following the link.

Note: pyrewton is still in development, and further functionalities are being added. Please see the GitHub repository for the latest developments.