cazy_webscraper

Release 2.0.0

Emma E. M. Hobbs

May 23, 2023

CONTENTS

For latest updates and development progress, please check the GitHub repository

CHAPTER

ONE

BUILD INFORMATION

CHAPTER

TWO

PYPI

CHAPTER

THREE

BIOCONDA

CHAPTER

FOUR

CAZY_WEBSCRAPER

`cazy_webscraper` is a Python3 package for the automated retrieval of Carbohydrate-Active enZyme (CAZyme) data from the CAZy database. This program is free to use under the MIT license, and we kindly request that, if you use this program or Python package, you cite it as indicated below.

> Hobbs, Emma E. M.; Pritchard, Leighton; Chapman, Sean; Gloster, Tracey M. (2021): cazy_webscraper Microbiology Society Annual Conference 2021 poster. figshare. Poster. https://doi.org/10.6084/m9.figshare.14370860.v7

`cazy_webscraper` retrieves data from CAZy, writing it to a local SQLite3 file (typically taking 10-15 minutes to scrape the entirety of CAZy).

Additionally, ``cazy_webscraper`` can:

- Retrieve the protein data from UniProt for CAZymes in the local database. This data includes:
  - UniProt accession
  - Protien name
  - Protein amino acid sequence
  - EC numbers
  - PDB accessions
- Retrieve protein sequences from NCBI GenBank for CAZymes in the local database.
- Write out protein sequences retrieved from UniProt and NCBI in FASTA format, and build a local BLAST database.
- Retrieve protein structures from the Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank, PDB, for CAZymes in the local database.
- Be configured to scrape the entire CAZy database, or recover only CAZymes filtered by user-supplied criteria, such as CAZy classes, CAZy (sub)family, or taxonomy.
- Retrieve the latest taxonomic classifications (including the complete lineage) from the NCBI Taxonomy database

CHAPTER

FIVE

QUICKSTART

We have produced a "Getting Started With `cazy_webscraper`" poster.

To download the entire CAZy dataset, and save the data set to the current working directory with the file name `cazy_webscraper_<date>_<time>.db`, use the following command structure:

```
cazy_webscraper <user_email>
```

Note: The user email address is a requirement of NCBI. NCBI is queried to identify the currect source organism for a given protein, when multiple source organisms are retrieved from CAZy for a single protein. For more information please see the NCBI Entrez documentation.

CHAPTER

SIX

COMMAND SUMMARY

Below are the list of commands (excluding required and optional arguments) included in `cazy_webscraper`.

CAZy

To retrieve data from CAZy and compile and SQLite database using `cazy_webscraper` command.

UniProt

To retrieve protein data from UniProt, use the `cw_get_uniprot_data` command.

The following data can be retrieved: - UniProt accession - Protein name - EC numbers - PDB accession - Protein sequences

GenBank

- To retrieve protein sequences from GenBank use the `cw_get_genbank_seqs` command.
- To retrieve the latest taxonomic classifications from NCBI Taxonomy using the `cw_get_ncbi_taxs` command.

Extract sequences

To extract GenBank and/or UniProt protein sequences from a local CAZyme database, use the `cw_extract_db_seqs` command.

PDB

To protein structure files from PDB use the `cw_get_pdb_structures` command.

NCBI taxonomies

Retrieve the latest taxonomic classifications (including the complete lineage from kingdom to strain) using the `cw_get_ncbi_taxs` command.

GTDB taxonomies

Retrieve the latest taxonomic classifications (incluidng the complete lineage from kingdom to strain) from the GTDB database using the `cw_get_gtdb_taxs` command.

Interrogate the database

To interrogate the database, use the `cw_query_database` command.

CHAPTER

SEVEN

BEST PRACTICE

When performing a series of many automated, repeated calls to a server it is polite to do this when internet traffic is lowest *at the server*. This is typically at the weekend and overnight.

When using `cazy_webscraper` to retrieve data from UniProt, NCBI or PDB, the webscraper can appear to run slowly but this may be due to bandwidth at the database server, or server speed. `cazy_webscraper` provides a progress bar to reassure the user that the webscraper is working.

Warning: Please **do not** perform a retrieval of UniProt, NCBI and/or PDB data for the entire CAZy dataset, unless absolutely unavoidable. Retrieving the data from any of these exteranl databases for the entire CAZy dataset will take several hours and may unintentionally deny the service to others.

CHAPTER

EIGHT

DOCUMENTATION

For details and updates on development, please consult the GitHub repository.

8.1 Installation

We support three ways to install `cazy_webscraper`.

- Using bioconda **[Recommended]**
- Using pip/PyPI
- Installation from source (git clone)

8.1.1 Requirements

- A POSIX-compliant operating system, e.g. Linux or MacOS.
- Python 3.8 or later
- An internet connection (to access CAZy and download data)

8.1.2 Installing with `Bioconda`

Tip: The most recent stable release of `cazy_webscraper` should always be avaiable from the `bioconda` channel.

To install `cazy_webscraper` using bioconda and all required packages, you can use the following command:

```
conda install -c bioconda cazy_webscraper
```

Note: If `conda` is not installed on your system, please see the `conda` website for instructions.

8.1.3 Installing with pip/PyPI

Tip: The most recent stable release of cazy_webscraper should always be avaiable from PyPI.

To install cazy_webscraper and all required packages using pip, you can use the following command:

```
python -m pip install cazy-webscraper
```

8.1.4 Installing from source

cazy_webscraper can be installed from the source code available at the GitHub repository.

Warning: The cazy_webscraper repository provides the development version of cazy_webscraper. This is the version that is most recently updated, but it may not be the latest stable version. In particular, the development version may contain features that are not yet in a stable version, and it may contain bugs.

Tip: To obtain the most recent *stable* source code from the cazy_webscraper repository, download a release from the releases page and extract the archive.

Attention: If you are using *conda*, you can use the Makefile in the cazy_webscraper repository to install cazy_webscraper from source, with the command make setup_env.

Obtaining cazy_webscraper source

The **development** version of cazy_webscraper is available at the GitHub repository and can be downloaded in the following ways.

To clone the cazy_webscraper repository using git, you can use the following command:

```
git clone https://github.com/HobnobMancer/cazy_webscraper
```

This will download the most recent development version of cazy_webscraper into a new directory called cazy_webscraper.

Alternatively, the **development** version can be downloaded as an archive file from the link below.. button:

```
* `source code .zip file <https://github.com/HobnobMancer/cazy_webscraper/archive/refs/
↪heads/master.zip>`_
```

This file can be extracted and will create the directory cazy_webscraper.

Required packages

cazy_webscraper requires several packages to be installed, in order to run. You can install these packages using the requirements files in the cazy_webscraper directory. The commands needed depend on your working environment.

Using pip/PyPI

All third-party packages can be installed using pip/PyPI:

```
pip install -r requirements.txt
pip install -r requirements-dev.txt  # only needed if you are developing the code
pip install -r requirements-pip.txt  # only needed if you are developing the code
```

Using conda

The conda package manager can be used to install all required packages for running cazy_webscraper, but the sphinx package is not available in conda and must be installed using pip:

```
conda install --file requirements.txt
conda install --file requirements-dev.txt  # only needed if you are developing the code
pip install -r requirements-pip.txt        # only needed if you are developing the code
```

Installing cazy_webscraper

The cazy_webscraper package can be installed from source using the following command, issued from the root directory of the cazy_webscraper repository:

```
python setup.py install
```

If you are intending to edit or develop the code, you can use the develop option instead of install:

```
pip install -e .
```

8.2 Quickstart

8.2.1 Installation

The most recent version of cazy_webscraper can be installed on your local machine using conda or pip. Both methods will install the cazy_webscraper command-line tool, and the Python package cazy_webscraper.

conda

```
conda install cazy_webscraper
```

pip

`pip` should distribute the latest version of `cazy_webscraper`, although there may be some minor lag between GitHub releases and `pip`.

```
pip3 install cazy_webscraper
```

Tip: `cazy_webscraper` can also be installed directly from source. More detailed, and alternative installation instructions can be found in the *Installation* section.

8.2.2 Getting Started Poster

For a quick summary of how to get started, check out our poster:

> Hobbs, Emma E. M.; Pritchard, Leighton; Gloster, Tracey M.; Chapman, Sean (2021): cazy_webscraper - getting started. FigShare. Poster. https://doi.org/10.6084/m9.figshare.14370869.v3

8.2.3 Getting Help From *cazy_webscraper*

To invoke the webscraper and get basic help, call the webscraper at the command line:

```
cazy_webscraper -h
```

The default behaviour of the scraper is:

- Scrape all entries in the CAZy database
- Write the resulting data to STDOUT
- Do not retrieve subfamilies (subfamily members will be retrieved but only their parent family be listed)
- Do not retrieve FASTA files from GenBank
- Do not retrieve protein sequences from PDB

8.2.4 Downloading a CAZy Family

To download the single CAZy family GH169, use the command:

```
cazy_webscraper --families GH169 -o GH169
```

This will create a new directory `GH169` in the current working directory, and will download all CAZy entries in the GH169 family to a new SQLite3 database in that directory:

8.3 Usage: Scraping CAZy

`cazy_webscraper` can be used to retrieve user-specified data sets from the CAZy database. The `cazy_webscraper` application can be invoked via the command line

8.3.1 Quick Start

To download the entire CAZy dataset, and save the data set to the current working directory with the final name `cazy_webscraper_<date>_<time>.db`, use the following command structure:

```
cazy_webscraper <user_email>
```

Note: The user email address is a requirement of NCBI. NCBI is queried to identify the currect source organism for a given protein, when multiple source organisms are retrieved from CAZy for a single protein. For more information please see the NCBI Entrez documentation.

Note: Typically, downloading the entire CAZy dataset takes 10-15 minutes, although this is dependent on the amount of avaible memory.

To print citation information (including the citations of third party tools used by `cazy_webscraper`):

```
cazy_webscraper --citation
```

Or

```
cazy_webscraper -C
```

To print version information (including the versions of third party tools used by `cazy_webscraper`):

```
cazy_webscraper --version
```

Or

```
cazy_webscraper -V
```

8.3.2 Command line options

Listed below are the required and optional command-line options when using `cazy_webscraper` to download data from CAZy.

`email` - **REQUIRED** User email address. This is required by NCBI Entrez for querying the Entrez server. - Not needed when printing out citation or version number information.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_data` - Path to a text file downloaded from CAZy containing a CAZy database dump

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--citation`, `-C` - Print the *cazy_webscraper* citation. When called, the program terminates after printng the citation and CAZy is **not** scraped.

`--classes` - list of classes from which all families are to be scrape.

`--database`, `-d` - Path to an **existings** local CAZyme database to add newly scraped too. Default: None.

`--db_output`, `-o` - Path to write out a **new** local CAZyme database to. Include the name of the new database, including the *.db* extension. Default: None.

Warning: Do not use ``–db_output`` and ``–database`` at the same time.

Note: If `--db_output` **and** `--database` are **not** called, `cazy_webscraper` will writes out a local CAZyme database to the cwd with the standardised name `cazy_webscraper_<date>_<time>.db`

`--delete_old_relationships` - Detele old CAZy family annotations of GenBank accessions. These are CAZy family annotations of a given GenBank accession are in the local database but the accession is not longer associated with those CAZy families, so delete old accession-family relationships.

`--families` - List of CAZy (sub)families to scrape.

`--force`, `-f` - force overwriting existing output file. Default: False.

Warning: If a specified output directory already exists, if `--force` is not called, `cazy_webscraper` will not overwrite the output and terminate.

`--genera` - List of genera to restrict the scrape to. Default: None, filter not applied to scrape.

`--kingdoms` - List of taxonomic kingdoms to restrict the scrape to. Default: None, filter is not applied.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete`, `-n` - When called, content in the existing output dir will **not** be deleted. Default: False (existing content is deleted).

Note: When the `--db_output` flag is used, `cazy_webscraper` will create any necessary parent directories. If the direct/immediate parent directory of the database exists, by default `cazy_webscraper` will delete the content in this parent directory.

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_log` - When called, content in the existing log dir will **not** be deleted. Default: False (existing content is deleted).

`--ncbi_batch_size` - The number of protein IDs submitted per batch to NCBI, when retrieving taxonomic classifications. Default 200.

`--retries`, `-r` - Define the number of times to retry making a connection to CAZy if the connection should fail. Default: 10.

`--skip_ncbi_tax` - Skip retrieving the latest taxonomic information for NCBI were multiple taxonomic classifications are retrieved from CAZy for a protein. The first taxonomy retrieved from CAZy will be added to the local CAZyme

database. Default False: will not retrieve taxon data from NCBI, will use the first taxon retrieved from the CAZy database dump.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--subfamilies`, `-s` - Enable retrival of CAZy subfamilies, otherwise **only** CAZy family annotations will be retrieved. Default: False.

`--species` - List of species written as Genus Species) to restrict the scraping of CAZymes to. CAZymes will be retrieved for **all** strains of each given species.

`--strains` - List of specific species strains to restrict the scraping of CAZymes to.

`--timeout`, `-t` - Connection timout limit (seconds). Default: 45.

`--validate`, - Retrieve CAZy family population sizes from the CAZy website and check against the number of family members added to the local CAZyme database, as a method for validating the complete retrieval of CAZy data.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

`--version`, `-V` - Print `cazy_webscraper` version number. When called and the version number is printed, `cazy_webscraper` is immediately terminated.

8.3.3 Basic Usage

The command-line options listed above can be used in combination to customise the scraping of CAZy. Some options (e.g. `--families` and `--classes`) define the broad group of data that will be scraped, others (e.g. `--species`) are used to filter and fine-tune the data that is scraped.

Defining CAZy families and classes to scrape

The 'definition' arguments (e.g. `--classes` and `--families`) indicate which groups of data will be selected for scraping from CAZy, e.g.

```
cazy_webscraper --families GH169 -o GH169.db
cazy_webscraper --classes AA -o AA.db
```

will download all CAZymes from the GH169 family, and the AA class, respectively. More than one class or family can be specified, e.g.

```
cazy_webscraper --families GH169,GH1,GH2,GH3 -o GH_families.db
cazy_webscraper --classes AA,CBM -o other_classes.db
```

and members of distinct families and classes can be selected simultaneously, e.g.

```
cazy_webscraper --families GH169,GH1,GH2,GH3 --classes AA,CBM -o complex_query.db
```

Note: CAZy families should be named using the standard CAZy syntax. GH1 is **accepted**. "gh1" and "Glycoside hydrolase 1" are **not** accepted.

Specifying output data location

By default `cazy_webscraper` writes out a SQL database file to the current working directory, with a filename with the following structure `cazy_webscraper_<date>_<time>.db`, where the date and time mark the time `cazy_webscraper` was called.

To specify the location of the output database the `--db_output` / `-o` option can be used:

```
cazy_webscraper --families GH169 -o GH169_output.db
```

will write an SQL database file to `GH169_output.db`.

If the target output file already exists, `cazy_webscraper` by default will not overwrite the existing file and will terminate. To overwrite an existing file use the `--force` / `-f` options:

```
cazy_webscraper --families GH169 -o GH169_output.db -f
```

A multi-layered path can be provided to `cazy_webscraper`. If any of the parent directories for the target output path do not exist, `cazy_webscraper` will build the necessary output direcotires. In the following command if the `cazy` and `families` directories do not exist, `cazy_webscraper` will build these directories:

```
cazy_webscraper --families GH169 -o cazy/families/GH169_output.db
```

If any of the output directories exist, by default, `cazy_webscraper` will terminate. To write to an existing output directory use the `--force` / `-f` options:

```
cazy_webscraper --families GH169 -o GH169_output.db -f
```

By default `cazy_webscraper` will delete the existing content in the existing output files. To not delete the content in the existing output directories use the `--nodelete` / `-n`:

```
cazy_webscraper --families GH169 -o GH169_output.db -f -n
```

If you already have an existing CAZy database, then specifying this database with the `-d` / `--database` option will cause the scraper to use the existing database rather than creating a new one:

```
cazy_webscraper --families GH169 -d GH169/GH169_output.db
```

Filtering CAZy families and classes

Options that apply a *filter* to restrict which CAZymes from a class or familiy are scraped from CAZy (e.g. `--families` and `--species`) may be applied in combination. For example:

```
cazy_webscraper --families GH169 \
    --species "Escherichia coli" \
    -o GH169_speciesEscherichia_coli.db
```

will download only the CAZymes in the GH169 family that are from the species *Escherichia coli*. The command:

```
cazy_webscraper --families PL14,PL15,PL16 \
    -o PL14_ec1.2.3.4_kingdomBacteria
```

will download only CAZymes in the PL14, PL15 and PL16 families that are from the kingdom *Bacteria*.

Note: `cazy_webscraper` input options can also be specified in a **YAML configuration file**, to enable transparency and reproducibility.

8.3.4 Configuration using a YAML file

All command-line options to control CAZy scraping can be provided instead *via* a YAML configuration file. This supports reproducible documentation of `cazy_webscraper` usage.

An template YAML file is provided in the `cazy_webscraper` repository (`scraper/scraper_config.yaml`):

```
# Under 'classes' list class from which all proteins will retrieved
# Under each families respective name, list the specific families/subfamilies to be␣
↪scraped
# Write the FULL family name, e.g. 'GH1', NOT only its number, e.g. '1'
# To list multiple families, each familiy must be on a new line starting indented once
# relative to the parent class name, and the name written within quotation marks.
# For more information on writing lists in Yaml please see:
# https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
classes:  # classes from which all proteins will be retrieved
Glycoside Hydrolases (GHs):
GlycosylTransferases (GTs):
Polysaccharide Lyases (PLs):
  - "PL28"
Carbohydrate Esterases (CEs):
Auxiliary Activities (AAs):
Carbohydrate-Binding Modules (CBMs):
genera:  # list genera to be scraped
 - "Trichoderma"
species:  # list species, this will scrape all strains under the species
strains:  # list specific strains to be scraped
kingdoms:  # Archaea, Bacteria, Eukaryota, Viruses, Unclassified
 - "Bacteria"
```

Attention: The YAML configuration file must contain all tags/headings indicated in the example configuration file found in the repository:

- classes
- Glycoside Hydrolases (GHs)
- GlycosylTransferases (GTs)
- Polysaccharide Lyases (PLs)
- Carbohydrate Esterases (CEs)
- Auxiliary Activities (AAs)
- Carbohydrate-Binding Modules (CBMs)
- genera
- species
- strains

- kingoms

Each value in the YAML mappings for these arguments must be listed on a separate line, indented by 4 spaces, and the class name encapsulated with single or double quotation marks. For example:

```
classes:
    - "GT"
    - "pl"
Glycoside Hydrolases (GHs):
    - "GH1"
    - "GH2"
```

Synonyms for CAZy classes

A number of synonyms may be provided for CAZy classes, e.g. both "GH" and "Glycoside-Hydrolases" are accepted as synonyms for "Glycoside Hydrolases (GHs)" (the name recorded at CAZy). These alternatives are defined in the `cazy_webscraper` repository, in the file `scraper/utilities/parse_configuration/cazy_dictionary.json`.

8.3.5 Scraping CAZy subfamilies

`cazy_webscraper` can scrape CAZy subfamilies, using the standard CAZy notation for subfamilies (e.g. `GH3_1`).

Note: If any subfamilies are specified for download/scraping in the YAML file, the command line argument `--subfamilies` must be used.

If a parent CAZy family is listed in the configuration file and `--subfamilies` is enabled at the command-line, all proteins catalogued under the named family and its subfamilies will be retrieved.

8.4 Tutorials on configuring `cazy_webscraper` to scrape CAZy

`cazy_webscraper` can be configured to retrieve user specified data sets from CAZy. The same configuration applies to the retrieval of protein data from UniProt, GenBank and PDB.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the scraping of CAZy. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` call the application using the command `cazy_webscraper` - this is the method used in this tutorial.

If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` using the command `python3 <path to cazy_scraper.py>`. For example, if you were located in the root of the repository, you would use: `python3 cazy_webscraper/cazy_scraper.py`.

8.4.1 Configuration via the command line

`cazy_webscraper` has only one required argument, the user email address. Therefore,
the scraper can be enabled to scrape all of CAZy using the following command:

```
cazy_webscraper myemail@domain.com
```

When no optional arguments are provided the default behaviour of the scraper will be performed. The default behaviour is to:

- Scrape the entirety of CAZy databases
- Write the resulting database to the current working directory
- Not retrieve subfamilies (members of subfamilies will be retrieved but only their parent family will be listed)

Note: For those new to using command-line tools: Arguments are additional pieces of information we add onto the end of the command. They are used to configure the specific behaviour performed by yjr computer when we tell it to perform a specific command. In the examples above the command is `cazy_webscraper myemail@domain.com`, where we have told to computer to run the Python program `cazy_webscraper` and submit the user email address to NCBI for the retrieval of source orgnaism data. No additional information was provided, the computer will invoke `cazy_webscraper` using its default behaviour. If you do not want the default behaviour of `cazy_webscraper` then we need to pass additionally arguments to the computer when telling it to run `cazy_webscraper`, which we cover in the section below.

8.4.2 Options configurable at the command line

The following behaviours of the `cazy_webscraper` can be configured at the command-line in the terminal:

- Limit the scraping of CAZy to specific CAZy classes, CAZy families, kingdoms, genuera, species, and/or strains
- Force writing out the database to a a new or existing directory
- Write out a log file of the packages operation
- Not delete content already present in the output directory
- Enable retrieving subfamilies
- Enable verbose logging during the operation of the webscraper

Here you can find a full list of the command-line flags and options.

How to use the command-line options

The command-line options listed above can be used in any combination to customise the scraping of CAZy. The options that apply a 'filter' to restrict which CAZymes are scraped from CAZy are applied in combination. For example, if the `--families` option and `--genera` option are called then only CAZymes from the specified families **and** belonging to source organisms within the defined genera will be retrieved.

We will now walk through some examples of how to use `cazy_webscraper`. All example code presumes `cazy_webscraper` was installled using `Bioconda` or `pip` and therefore, be simply called using the command `cazy_webscraper`.

Note: For those new to using command-line tools: flags Command-line flags are used to tell the computer specifically which option(s) to change. Flags **always** come after the command.

The abbreivated version of a flag is given the prefixed of a single dash, followed by a single letter. For example, `-s`, `-o` and `-l` are all examples of short hand flags.

The long version of a flag is prefixed by two dashes, followed by complete words. For example, `--output` is the long version of the `-o`.

The flags used by a program are defined within the program. This means the flag `-o` may represent different options for different programs. Always make sure to check the documentation to see what flags are provided with the program, and what they do!

You can use the command-line to list all flags for a program/tool by typing in the command to invoke that tool, followed by the flag `--help` or `-h`. For example: `cazy_webscraper --help`.

8.4.3 Configuring were the output is saved

Creating a new database

Instead of writing out the database to the current working directory using the default database name (`cazy_webscraper_<date>_<time>.db`), we can name the database and directory that the database created by `cazy_webscraper` is written to by calling the `--output` flag.

We add the flag to the command that invokes `cazy_webscraper`. For example, to write the output to the directory 'cazyme_database' with the file name 'cazyme_database.db' we can use:

```
cazy_webscraper --output cazyme_database/cazyme_database.db
```

OR we can use the short hand version of the `--output` flag, `-o`:

```
cazy_webscraper -o cazyme_database/cazyme_database.db
```

Note: The final element of the path provided after the `--output` / `-o` flag is the name of database compiled by `cazy_webscraper`.

The output directory does not have to exist when `cazy_webscraper` is invoked. `cazy_webscraper` can make the output directory, including all necessary parent directories.

The `--output` flag can take an infinetly long path. For example, we could use:

```
cazy_webscraper -o data/cazyme_research/cazyme_database/cazyme_database.db
```

If the directories `cazymes_research` and `cazyme_database` did not exist, then `cazy_webscraper` will build these for you.

Overwriting an existing database or directory

If you want to write the output CAZyme database to a directory and/or file that already exists, you will need to add the force (`--force` *or* `-f`) flag anywhere to the `cazy_webscraper` command. For example:

```
cazy_webscraper -o data/cazyme_research/cazyme_database/cazyme_database.db -f
```

By default `cazy_webscraper` will delete all content in an already existing output directory. Therefore, in the above example, if the directory `cazyme_database` already existed, `cazy_webscraper` would delete all content in the directory then proceed.

You may wish to retain the data already in that directory. To do this add the 'no delete' (`--nodelete` *or* `-n`) flag anywhere to the `cazy_webscraper` command. For example:

```
cazy_webscraper -o data/cazyme_research/cazyme_database/cazyme_database.db -f -n
```

The order you invoke *any* of the optional flags does not matter, for example the following three examples perform the exact same operation as the code given above:

```
cazy_webscraper --force -o data/cazyme_research/cazyme_database/cazyme_database.db -f
```

```
cazy_webscraper -n -o data/cazyme_research/cazyme_database/cazyme_database.db -f
```

```
cazy_webscraper --nodelete --force --output data/cazyme_research/cazyme_database/cazyme_
↪database.db
```

The above examples also highlight that it does not matter if you use the long or short versions of each of the flags.

Note: If you elect to write the database to a file in the current working directory, you do not need to worry about `cazy_webscraper` deleting content in the current working directory. This only applies if you chose to write the database to a directory over than the current working directory.

Add the scraped data to an existing CAZyme database

You may wish to scrape CAZy in multiple stages; maybe your internet dropped out while scraping CAZy and you don't want to start again, or maybe you scraped CAZy but missed out a species of interest. No matter the reason `cazy_webscraper` allows you to add more CAZyme data to an existing database previously created by `cazy_webscraper`.

To do this add the database (`--database` or `-d`) flag to the `cazy_webscraper` command, followed by the path to the CAZyme database you want to add your scraped CAZy data to. For example, to add data to an existing database in `cazy/cazyme_db.db` use the command:

```
cazy_webscraper -- database cazy/cazyme_db.db
```

Note: Don't forget the .db file extension at the end of the path!

All the paths we pass to `cazy_webscraper` are a *relative* path. This means `cazy_webscraper` will start in the directory the terminal is currently pointed out, and follow the path from there. For example, if we used the command:

```
cazy_webscraper -d my_cazyme_databases/my_cazyme_database.db
```

Then the computer will look for a directory called `my_cazyme_databases` in the directory the terminal is looking at, then within the `my_cazyme_databases` directory the computer will look for the file `my_cazyme_database.db`.

8.4.4 Specifying CAZy classes and families to scrape

Scraping specific classes

If instead of scraping all of CAZy, you want to only scrape CAZymes from specific CAZy classes then add the `--classes` flag followed by the classes you want to scrape. If you want to list multiple classes, separate the classes with a single comma. When you specify a CAZy class to scrape, *all* CAZy families within that class will be scraped.

For example, if you want to scrape all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cazy_webscraper --classes Glycoside Hydrolases,Carbohydrate Esterases
```

`cazy_webscraper` excepts multiple synonyms for each CAZy class:

- **Glycoside Hydrolases (GHs):** Glycoside-Hydrolases, Glycoside-Hydrolases, Glycoside_Hydrolases, GlycosideHydrolases, GLYCOSIDE-HYDROLASES, GLYCOSIDE-HYDROLASES, GLYCOSIDE_HYDROLASES, GLYCOSIDEHYDROLASES, glycoside-hydrolases, glycoside-hydrolases, glycoside_hydrolases, glycosidehydrolases, GH, gh
- **GlycosylTransferases (GTs):** Glycosyl-Transferases, GlycosylTransferases, Glycosyl_Transferases, Glycosyl Transferases, GLYCOSYL-TRANSFERASES, GLYCOSYLTRANSFERASES, GLYCOSYL_TRANSFERASES, GLYCOSYL TRANSFERASES, glycosyl-transferases, glycosyltransferases, glycosyl_transferases, glycosyl transferases, GT, gt
- **Polysaccharide Lyases (PLs):** Polysaccharide Lyases, Polysaccharide-Lyases, Polysaccharide_Lyases, PolysaccharideLyases, POLYSACCHARIDE LYASES, POLYSACCHARIDE-LYASES, POLYSACCHARIDE_LYASES, POLYSACCHARIDELYASES, polysaccharide lyases, polysaccharide-lyases, polysaccharide_lyases, polysaccharidelyases, PL, pl
- **Carbohydrate Esterases (CEs):** Carbohydrate Esterases, Carbohydrate-Esterases, Carbohydrate_Esterases, CarbohydrateEsterases, CARBOHYDRATE ESTERASES, CARBOHYDRATE-ESTERASES, CARBOHYDRATE_ESTERASES, CARBOHYDRATEESTERASES, carbohydrate esterases, carbohydrate-esterases, carbohydrate_esterases, carbohydrateesterases, CE, ce
- **Auxiliary Activities (AAs):** Auxiliary Activities, Auxiliary-Activities, Auxiliary_Activities, AuxiliaryActivities, AUXILIARY ACTIVITIES, AUXILIARY-ACTIVITIES, AUXILIARY_ACTIVITIES, AUXILIARYACTIVITIES, auxiliary activities, auxiliary-activities, auxiliary_activities, auxiliaryactivities, AA, aa
- **Carbohydrate-Binding Modules (CBMs):** Carbohydrate-Binding-Modules, Carbohydrate_Binding_Modules, Carbohydrate_Binding Modules, CarbohydrateBindingModules, CARBOHYDRATE-BINDING-MODULES, CARBOHYDRATE_BINDING_MODULES, CARBOHYDRATE_BINDING MODULES, CARBOHYDRATEBINDINGMODULES, carbohydrate-binding-modules, carbohydrate_binding_modules, carbohydrate_binding modules, carbohydratebindingmodules, CBMs, CBM, cbms, cbm

---

Tip: These synonyms are stored in a JSON found at `scraper/utilities/parse_configuration/cazy_dictionary.json`. Storing these synonyms allows you to modify this file if you wish to add your own synonoms for each CAZy class.

If you have your own synonyms these can be used by using the `--cazy_synonyms` flag, followed by the path to your JSON file. This JSON file **must** have the same architecture as the JSON filed used by `cazy_webscraper`.

Scraping specific families

To specify specific CAZy families to scrape, add the `--families` flag followed by the families you want to scrape. If you want to scrape multiple families, list all the families you wish to scrape, with each family separated with a single comma.

For example, if you want to scrape all CAZymes from GH2, PL5, CE1, CE2 and AA10 use:

```
cazy_webscraper --families GH2,PL5,CE1,CE2,AA10
```

> **Warning:** Make sure to use the accepted CAZy nomenclature; 'GH2' is accepted but 'gh2' is not.

Scraping specific classes AND families

If you want to specify specific CAZy classes *and* families to scrape then add *both* the `--classess` *and* `-families` flags, because you can combine, mix-and-match, any combination of optional flags when invoking `cazy_webscraper`.

For example, if we wanted to scrape all CAZymes from GH1, PL9 and *all* of CE we would use the command:

```
cazy_webscraper --families GH1,PL9 --classes CE
```

It does **not** matter what order you add the optional flags to your command. Therefore, if we wanted to scrape all CAZymes from PL1, PL2, PL3 and *all* of GH and CE, both:

```
cazy_webscraper --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cazy_webscraper --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

> **Note:** In the example `cazy_webscraper --classes GH,CE --families PL1,PL2,PL3` all CAZymes from PL1, PL2 and PL3 would be retrieved, but no CAZymes from the other PL families, in addition all CAZymes from all GH and CE families would be retrieved, but no CAZymes from AA, GT or CBM families would be retrieved.

8.4.5 Applying taxonomic

Specifying kingdoms

You may only be interest in CAZymes that are derived from species from a specific taxonomic kingdom. CAZy classifies source organisms under one of 5 kingdoms:

- Archaea
- Bacteria
- Eukaryota
- Viruses
- Unclassified

To restrict the scraping of CAZy to retrieve CAZymes only derived from species from specific taxonomic kingdoms add the `--kingdoms` flag to the `cazy_webscraper` command followed by the kingdoms to limit the retrieval of CAZymes to. To list multiple kingdoms you need only add the `--kingdoms` flag *once*, then list all the kingdoms you are interested in, separated by a single comma.

For example, if you want to retrieve CAZymes only from bacterial and eukaryotic species then use the command

```
cazy_webscraper --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

Speciying Genera to scrape

You can customise the scraping of CAZy to retrieve only CAZymes from *all* species from specific genera. To do this add the `--genera` flag to the `cazy_webscraper` command followed by your genera of interes.

To list multiple genera, you need to only add the `--genera` flag *once* followed by a list of all your genera, with each genera separated with a single comma and *no* spaces.

For example, if we wanted to retrieve all CAZymes from *all* Aspergillus, Trichoderma and Streptomyces species we would use the command:

```
cazy_webscraper --genera Aspergillus,Trichoderma,Streptomyces
```

Note: The order that the genera are listed does **not** matter.

Warning: Make sure to use the expect practise for writing genera names, each genus starts with a **captial** letter and all other letters are lower case.

Aspergillus is **correct**

asepergillus is **incorrect**

ASPERGILLUS is **incorrect**

Specifying species of organisms to scrape

You can specify to retrieve only CAZymes derived from specific species. To do this add the `--species` flag to the `cazy_webscraper` command, followed by a list of all species you wish to retrist the retrieval of CAZymes to. Separate each species with a single comma. Also for each species use the full scientific name for the species.

For example, if we wanted to retrieve all CAZymes from *Aspergillus niger* and *Aspergillus fumigatus* we would use the command:

```
cazy_webscraper --species "Aspergillus niger,Asepergillus fumigatus"
```

Note: The order that the species are listed does **not** matter, and separate multiple species names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When providing any parameter that contains a space within it, enclose the entire parameter is single or double quotation marks

"Aspergillus niger,Trichoderma reesie" is **correct**

'Aspergillus niger,Trichoderma reesie' is **correct**

asepergillus niger,Trichoderma reesie is **incorrect**

Therefore, when using the `-species` and `--strains` flag,s **always** enclode the provided argument (or parameter) in single or double quotation marks.

Warning: When you specify a species `cazy_webscraper` will retrieval CAZymes from *all* strains of the species.

Specify specific strains of species to scrape

You may only be interested in specific strains of a species. Instead of scraping CAZymes for all strains of a given speices, add the `--strains` flag followed by the specific species strains you wish to restrict the retrieval of CAZymes to.

List the full scientific name followed by the strain name. To specify multiple strains, list all strains of interest and separate with a single comma with **no** space.

For example, if we wanted to retrieve all CAZymes from Aspergillus niger ATCC 1015 and Aspergillus uvarum CBS 121591 we would use the command:

```
cazy_webscraper --strains "Aspergillus niger ATCC 1015,Aspergillus uvarum CBS 121591"
```

he order that the strains are listed does **not** matter.

Note: If you use the `--species`, `--genera` and `--strains` flags in any combination and a source organism matches multiple of the taxonomy critera, the CAZymes derived from that species will only be retrieved **once**.

For example, using the command `cazy_webscraper --genera Aspergillus --species "Aspergillus niger" --strains "Aspergillus niger ATCC 1015"` will retrieve all CAZymes from *all* Aspergillus species *once*.

The higher taxonomy levels take president, and the command will not retrieve all CAZymes from all Aspergillus species once AND all CAZymes from Aspergillus niger strains as well, and then retrieve another copy of all CAZymes from Aspergillus niger ATCC 1015.

Warning: When providing any parameter that contains a space within it, enclose the entire parameter is single or double quotation marks

> "Aspergillus niger,Trichoderma reesie" is **correct**
>
> 'Aspergillus niger,Trichoderma reesie' is **correct**
>
> asepergillus niger,Trichoderma reesie is **incorrect**
>
> Therefore, when using the `-species` and `--strains` flag,s **always** enclode the provided argument (or parameter) in single or double quotation marks.

Combining taxonomic filters

You can combine any combination of `cazy_webscraper` optional flags, including combining the taxonomic filtersFor example, you may wish to retrieve all CAZyme derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cazy_webscraper` command. The command we would use would be:

```
cazy_webscraper --kingdoms viruses --genera Aspergillus --species "Layia carnosa,Layia␣
↪chrysanthemoides" --strains "Trichoderma reesei QM6a,Trichoderma reesei QM9414"
```

Note: This is a single command written on a single line. When typing the command into the terminal do not fit enter until you have finished the command.

Warning: If you use the `--species`, `--genera` and `--strains` flags in any combination and a source organism matches multiple of the taxonomy critera, the CAZymes derived from that species will only be retrieved **once**. For example, using the command `cazy_webscraper --genera Aspergillus --species Aspergillus niger --strains Aspergillus niger ATCC 1015` will retrieve all CAZymes from *all* Aspergillus species *once*.

When combining taxonomy filters, the higher taxonomy levels take president. For example, the :command:

```
cazy_webscraper --genera Aspergillus --species "Aspergillus niger" --strains
↪"Aspergillus niger ATCC 1015"
```

will not retrieve all CAZymes from all Aspergillus species once AND all CAZymes from Aspergillus niger strains as well. `cazy_webscraper` will retrieval all CAZymes for all strains of *Aspergillus niger*.

8.4.6 Enabling retrieving subfamily annotations

By default `cazy_webscraper` only retrieves the CAZy family annotation for each protein, it does not retrieve the CAZy subfamily annotation. For example, a CAZyme within the CAZy subfamily GH3_1, will be stored in the local CAZyme database as only a GH3 CAZyme.

To retrieve the CAZy family **and** CAZy subfamily annotations, add the `-subfamilies` / `-s` flag, anywhere in the `cazy_webscraper` command. For example:

```
cazy_webscraper --families GH3 --subfamilies
```

This command will retrieve all CAZymes from GH3, and will retrieve the CAZy family **and** CAZy subfamily annotations. For example, a CAZyme in CAZy subfamily GH3_1 will be stored in the local database under the CAZy family GH3 and the CAZy subfamily GH3_1.

8.4.7 Combining CAZy class, CAZy family and taxonomy filters

You can use any combination of the CAZy class, CAZy family and taxonomy filters to fully customise the scrape of CAZy.

Below are some examples:

Example 1 To retrieve all CAZymes from all CBM families, GH1, GH2 and PL9, and that are derived from any Aspergillus species:

```
cazy_webscraper --classes CBM --families GH1,GH2,PL9 --genera Aspergillus
```

Example 2 To retrieve all CAZymes from GH1, and GH2 that are derived from any bacterial species:

```
cazy_webscraper --families GH1,GH2 --kingdoms bacteria
```

Example 3 To retrieve CAZymes from all viral species, and all Aspergillus niger strains which are catalogued within GH3_1 and GH3_2

```
cazy_webscraper --families GH3_1,GH3_2 --subfamilies --species "Aspergillus niger" --
↪kingdoms Bacteria
```

8.4.8 Configuration file

Whenever `cazy_webscraper` is invoked and adds data to a database, the configuration of `cazy_webscraper` (this is the kingdoms, genera, species, strains, CAZy classes and CAZy family filters which were applied) and the data and time the scrape was initiated is logged in the database. However, for optimal reproduction of how `cazy_webscraper` was used in your research, you can create shareable documentation that others can use to reproduce your CAZy dateset. This is achieved by creating a configuration file rather than configuring the performance of `cazy_webscraper` at the command line.

Creating a configuration file

`cazy_webscraper` uses the YAML file type for its configuraiton file; if you are new to YAML files please find more detailed information on YAML files [here](https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html).

A template and example configuration file for scrapping CAZy using `cazy_webscraper` can be found in the repo, in the `configuration_files` directory.

The configuration YAML **must** contain the following tags/headings (identical to how they are presented below):

- classes
- Glycoside Hydrolases (GHs)
- GlycosylTransferases (GTs)
- Polysaccharide Lyases (PLs)
- Carbohydrate Esterases (CEs)
- Auxiliary Activities (AAs)
- Carbohydrate-Binding Modules (CBMs)
- genera
- species
- strains
- kingdoms

Note: The order of the tags/headings does not matter.

Scraping specific CAZy classes

Under the **classes** heading list any classes to be scrapped. For each CAZy class listed under 'classes', CAZymes will be retrieved for every CAZy family within the CAZy class.

Each class must be listed on a separate line, indented by 4 spaces, and the class name encapsulated with single or double quotation marks. For example:

```
classes:
    - "GH"
    - "PL"
```

The same CAZy class name synonyms used for the command line are accepted for the configuration file.

Scraping specific CAZy families

Under the each of the class names listed in the configuration file, list the names of specific **families** to be scraped from that class. The respective classes of the specificed families do **not** need to be added to the 'classes' list.

Write the true name of the family not only it's number, for example **GH1** is excepted by **1** is not.

Name families using the standard CAZy nomenclature, such as **"GT2"** and **NOT "GlycosylTransferases_2"**. Additionally, use the standard CAZy notation for subfamilies (**GH3_1**).

> **Warning:** If any subfamilies are listed within the configuration file, the retrieval of subfamilies **must** be enabled at the command line uisng `--subfamilies`.

Each family must be listed on a separate line and the name surrounded by double or single quotation marks. For example:

```
Glycoside Hydrolases (GHs):
    - "GH1"
    - "GH2"
    - "GH3_1"
```

Example configuration file

Below is an example of the content you may wish to put in a configuration file. Using this file will retrieve all CAZymes in CAZy class AA, CAZy families GH1, GH3 and PL9 that are either derived from a bacterial or *Trichoderma* species.

```
classes:
   - "AA"
Glycoside Hydrolases (GHs):
   - "GH1"
   - "GH3"
GlycosylTransferases (GTs):
Polysaccharide Lyases (PLs):
   - "PL9"
Carbohydrate Esterases (CEs):
Auxiliary Activities (AAs):
Carbohydrate-Binding Modules (CBMs):
genera:
   - "Trichoderma"
species:
strains:
kingdoms:
   - "Bacteria"
```

Note: Indentations consist of 4 spaces.

You can add 'comments' to configuration file. Comments are section of text that are not read by `cazy_webscraper` and allow you to add notes to your configuration file. For example:

```
# This is a comment, text following a hashtag '#' on the same line is not read by cazy_
↪webscraper
```

(continues on next page)

(continued from previous page)

```
# https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
classes:  # classes from which all proteins will be retrieved
Glycoside Hydrolases (GHs):  # include two spaces between the end of the code and the␣
↪hashtag
GlycosylTransferases (GTs):
Polysaccharide Lyases (PLs):
- "PL28"
Carbohydrate Esterases (CEs):
Auxiliary Activities (AAs):
Carbohydrate-Binding Modules (CBMs):
genera:  # list genera to be scraped
- "Trichoderma"
species:  # list species, this will scrape all strains under the species
strains:  # list specific strains to be scraped
kingdoms:  # Archaea, Bacteria, Eukaryota, Viruses, Unclassified
- "Bacteria"
ECs:  # only CAZymes with at least one of these EC numbers will be scrapped
```

Example configuration files and tempate files can be found hre.

Using a configuration file

Once you have created a configuration file (we recommend modifying the template one provided with `cazy_webscraper` you then need to invoke `cazy_webscraper` and tell it you are using a configuration file. To do this we add the `--config` / `-c` flag to the `cazy_webscraper` command, followed by the path to the configuration file.

The path we pass to `cazy_webscraper` is a *relative* path. This means `cazy_webscraper` will start in the directory the terminal is currently pointed out, and follow the path from there. For example, if we used the command:

```
cazy_webscraper -c scraper/scraper_config.yaml
```

Then the computer will look for a directory called `scraper` in the directory the terminal is looking at, then look within the `scraper` directory for a yaml file called `scraper_config.yaml`.

Note: To check which directory `cazy_webscraper` is pointed at type `pwd` into the terminal and hit enter. This is the 'Present Working Directory' command, which will print the path to the directory the terminal is presently looking at.

Warning: Your path must point directly to the YAML file. Don't forget the '.yaml' file extension!

8.4.9 Using a configuration and the command-line

You can configure `cazy_webscraper` using a combination of command line arguments and a configuration file.

If a CAZyme matches at least one of the configuration data (whether if be from the terminal of the configuration file), one copy of the CAZyme record will be added to the SQL database, and only **one copy**, no matter how many of the configuration data the CAZyme meets.

To use a configuration file and a the command-line to configure `cazy_webscraper`, use the configuration file `--config` flag followed by the path to the configuration file and any of the additional optional flags you wish to use.

Note: The order you invoke the optional flags **does not** matter.

8.4.10 Additional operations to fine tune how cazy_webscraper operates

Scraping data from a previously downloaded CAZy txt file

CAZy provides access to data within its database via text files. cazy_webscraper downloads the CAZy text file containing all data within the CAZy database, providing a database dump. This file is then written to the cache directory (by default, called .cazy_webscraper_<date>_<time>).

For consistency in the dataset, you may wish to perform multiple scrapes of CAZyme data from the same CAZy text file. This could be a CAZy text file you have downloaded from CAZy or a text file downloaded by cazy_webscrapper.

To direct cazy_webscraper to retrieve CAZyme data from a previously downloaded CAZy text file, using the --cazy_data flag, followed by the path to the text file. For example:

```
cazy_webscraper --cazy_data cazy_db/cazy_data.txt
```

Warning: --cazy_data must be pointed directly at the text file, **not** a zipped file containing the CAZy data text file.

Writing out a log file

If you want to have a log file of all terminal output produced by cazy_webscraper then add the log --log / -l anywhere to the cazy_webscraper command, followed by a path to write the log file to. This path is a *relative* path and must include target a log file specifically. For example:

```
cazy_webscraper --subfamilies --genera Aspergillus --log log_dir/cazy_webscraper_log.log
```

Warning: The log file does not already have to exist for cazy_webscraper to write to it; however, all directories included in the path must already exist.

Verbose logging

For more detailed logging (which includes not only error and warning messages (the default) but also configuration setup, number of proteins retrieved etc.), add the verbose logging flag (--verbose or -v) anywhere to the cazy_webscraper command. For example:

```
cazy_webscraper --subfamilies --genera Aspergillus -v
```

The verbose flag can be used in combination with the log flag to write all terminal output to a log file.

Changing the connection timeout limit

Sometimes the connection to the CAZy server times out. By default if a connection is attempted to made to CAZy and no response is recieved within 45 seconds, then `cazy_webscraper` interprets this as the connection timing out. `cazy_webscraper` then waits 10 seconds and retries the connection.

You can change how long the computer waits for a response from the CAZy server before classifying the connection as timed out by adding the `--timeout` flag to the `cazy_webscraper` command, followed by the number of seconds you want the computer to wait for a response from CAZy before classifying the connection as timing out.

For example, to set the connection timeout limit to 30 seconds use the command:

```
cazy_webscraper --timeout 30
```

The timeout flag can be used in combination with other flags, for example:

```
cazy_webscraper --subfamilies --genera Aspergillus -v --timeout 30
```

8.5 The Local CAZyme Database Structure

To facilitate the thorough interrogation of data retrieved from CAZy and minimise storing duplicate and redundant data, data retrieved from CAZy is stored in a local SQL database. Every CAZyme scraped from CAZy has the following data:

- Protein name
- CAZy (sub)family
- GenBank accession(s)

Each CAZyme may or may not have the following data, depending on the entry:

- EC number(s)
- UniProt acession(s)
- PDB accession(s)

Note: EC numbers, UniProt accessions and PDB accessions can be retrieved from UniProt for CAZymes in the local CAZyme database using `cazy_webscraper`.

8.5.1 Database Schema

Below is the database ORM. It plots the relationships between elements in each table.

The database ORM (schema) can also be viewed Here

8.5.2 Retrieve the schema of a local CAZyme database

The schema of a local CAZyme database can be retrieved using `cazy_webscraper`:

Alternatively, *sqlite3* can be used to retrieve the schema:

8.5.3 The Logs table

The database built by `cazy_webscraper` contains a table called 'Logs'. This table logs every scrape of CAZy, UniProt and GenBank which added data to the database.

The table contains the following columns and data:

- **log_id:** Autoincrement ID number
- **date:** Date scrape was initated (in ISO format)
- **time:** Time scrape was initated (in ISO format)
- **database:** Name of the external database from which data was retrieved (i.e. 'CAZy', 'UniProt' or 'GenBank')
- **retrieved_annotations:** List of annotation types retrieved (e.g. 'EC number, PDB accession, Sequence')
- **classes:** CAZy classes for which data was retrieved
- **families:** CAZy families for which data was retrievedclasses
- **kingdoms:** taxonomy Kingdoms filteres applied
- **genera_filter:** taxonomy Kingdoms filteres applied
- **species_filter:** taxonomy Kingdoms filteres applied
- **strains_filter:** taxonomy Kingdoms filteres applied
- **ec_filter:** EC fliters applied to retrieve data for CAZymes annotated with the specified EC numbers (only applies to retrieval of data from UniProt, GenBank and PDB)
- **cmd_line:** Reproduction of the command line arguments passed to `cazy_webscraper`.

The 'Logs' table allows any one who uses the database to see how the dataset was compiled.

8.5.4 Genbanks

The Genbanks table contains data retrieved from CAZy and NCBI GenBank. From CAZy the GenBank protein accession. The protein sequence for the protein can be retrieved from NCBI GenBank using `cazy_webscraper`, and stored in the Genbanks table, as well as the date the sequence was updated in NCBI. The sequence update date is used to check if a newly retrieved protein sequence from GenBank is newer than the sequence stored in the database, and the sequence in the database should be updated.

8.5.5 Genbanks_CazyFamilies

The Genbanks_CazyFamilies table is a relationship. The table defines which protein is assigned to which CAZy family.

8.5.6 CazyFamilies

The CazyFamilies lists of CAZy families retrieved from CAZy. If CAZy subfamilies are retrieved each CAZy subfamily is associated with its parent CAZy family.

8.5.7 Taxs

The Taxs table stores taxonomy database, storing the genus and species of the source organisms of CAZymes retrieved from CAZy. Each source organism is associated with a taxonomic class.

8.5.8 Kingdoms

The Kingdoms table lists all taxonomic kingdoms of the source organisms downloaded from CAZy.

8.5.9 UniProts

The UniProts table contains protein data retrieved from UniProt using `cazy_webscraper`. This includes:

- UniProt ID
- Protein name
- Protein sequence
- Date the protein sequence was last upated in UniProt

8.5.10 Genbanks_Ecs

The Genbanks_Ecs table is a relationship table, and defines which proteins are annotated with which EC numbers.

8.5.11 Ecs

The Ecs table lists all EC numbers retrieved from UniProt using `cazy_webscraper`. The EC numbers stored in the local CAZome database are do not have the 'EC' prefix.

8.5.12 Genbanks_Pdbs

The Genbanks_Pdbs is a relationship table, and defines which PDB accessions belong to which protein.

8.5.13 Pdbs

The Pdbs table contains all PDB accessions retrieved from UniProt using `cazy_webscraper`.

Note: Not all PDB accessions represented in a CAZyme record at CAZy are necessarily present in PDB. For example, some accessions are placeholders while structures are under embargo.

Note: PDB/RCSB protein structures are not recorded in the local SQLite3 database. They are written to disk in a user-specified directory.

8.5.14 The Schema

As of *cazy_webscraper* version >= 2.3.0, the schema of a local CAZyme database will be:

```
CREATE TABLE IF NOT EXISTS "Kingdoms" (
        kingdom_id INTEGER NOT NULL,
        kingdom VARCHAR,
        PRIMARY KEY (kingdom_id),
        UNIQUE (kingdom)
);
CREATE TABLE IF NOT EXISTS "GtdbTaxs" (
        gtdb_tax_id INTEGER NOT NULL,
        kingdom VARCHAR,
        phylum VARCHAR,
        tax_class VARCHAR,
        tax_order VARCHAR,
        family VARCHAR,
        genus VARCHAR,
        species VARCHAR,
        release VARCHAR,
        PRIMARY KEY (gtdb_tax_id),
        UNIQUE (kingdom, phylum, tax_class, tax_order, family, genus, species, release)
);
CREATE TABLE IF NOT EXISTS "CazyFamilies" (
        family_id INTEGER NOT NULL,
        family VARCHAR NOT NULL,
        subfamily VARCHAR,
        PRIMARY KEY (family_id),
        UNIQUE (family, subfamily)
);
CREATE INDEX fam_index ON "CazyFamilies" (family, subfamily);
CREATE TABLE IF NOT EXISTS "NcbiTaxs" (
        ncbi_tax_id INTEGER NOT NULL,
        kingdom VARCHAR,
        phylum VARCHAR,
        tax_class VARCHAR,
        tax_order VARCHAR,
        family VARCHAR,
        genus VARCHAR,
```

(continues on next page)

(continued from previous page)

```
        species VARCHAR,
        strain VARCHAR,
        PRIMARY KEY (ncbi_tax_id),
        UNIQUE (ncbi_tax_id)
);
CREATE INDEX ncbi_index ON "NcbiTaxs" (ncbi_tax_id, genus, species);
CREATE TABLE IF NOT EXISTS "Uniprots" (
        uniprot_id INTEGER NOT NULL,
        uniprot_accession VARCHAR,
        uniprot_name VARCHAR,
        sequence VARCHAR,
        seq_update_date VARCHAR,
        PRIMARY KEY (uniprot_id),
        UNIQUE (uniprot_accession)
);
CREATE INDEX uniprot_option ON "Uniprots" (uniprot_id, uniprot_accession);
CREATE TABLE IF NOT EXISTS "Ecs" (
        ec_id INTEGER NOT NULL,
        ec_number VARCHAR,
        PRIMARY KEY (ec_id),
        UNIQUE (ec_number)
);
CREATE INDEX "ix_Ecs_ec_number" ON "Ecs" (ec_number);
CREATE TABLE IF NOT EXISTS "Pdbs" (
        pdb_id INTEGER NOT NULL,
        pdb_accession VARCHAR,
        PRIMARY KEY (pdb_id),
        UNIQUE (pdb_accession)
);
CREATE INDEX pdb_idx ON "Pdbs" (pdb_accession);
CREATE TABLE IF NOT EXISTS "Logs" (
        log_id INTEGER NOT NULL,
        date VARCHAR,
        time VARCHAR,
        "database" VARCHAR,
        retrieved_annotations VARCHAR,
        classes VARCHAR,
        families VARCHAR,
        kingdoms VARCHAR,
        genera_filter VARCHAR,
        species_filter VARCHAR,
        strains_filter VARCHAR,
        ec_filter VARCHAR,
        cmd_line VARCHAR,
        PRIMARY KEY (log_id)
);
CREATE TABLE IF NOT EXISTS "Taxs" (
        taxonomy_id INTEGER NOT NULL,
        genus VARCHAR,
        species VARCHAR,
        kingdom_id INTEGER,
        PRIMARY KEY (taxonomy_id),
```

(continues on next page)

(continued from previous page)

```
        UNIQUE (genus, species),
        FOREIGN KEY(kingdom_id) REFERENCES "Kingdoms" (kingdom_id)
);
CREATE INDEX organism_option ON "Taxs" (taxonomy_id, genus, species);
CREATE TABLE IF NOT EXISTS "Genomes" (
        genome_id INTEGER NOT NULL,
        assembly_name VARCHAR,
        gbk_version_accession VARCHAR,
        gbk_ncbi_id INTEGER,
        refseq_version_accession VARCHAR,
        refseq_ncbi_id INTEGER,
        gtdb_tax_id INTEGER,
        PRIMARY KEY (genome_id),
        UNIQUE (assembly_name, gbk_version_accession, refseq_version_accession),
        FOREIGN KEY(gtdb_tax_id) REFERENCES "GtdbTaxs" (gtdb_tax_id)
);
CREATE INDEX genome_options ON "Genomes" (assembly_name, gbk_version_accession, refseq_
↪version_accession);
CREATE TABLE IF NOT EXISTS "Genbanks" (
        genbank_id INTEGER NOT NULL,
        genbank_accession VARCHAR,
        sequence VARCHAR,
        seq_update_date VARCHAR,
        taxonomy_id INTEGER,
        ncbi_tax_id INTEGER,
        uniprot_id INTEGER,
        PRIMARY KEY (genbank_id),
        UNIQUE (genbank_accession),
        FOREIGN KEY(taxonomy_id) REFERENCES "Taxs" (taxonomy_id),
        FOREIGN KEY(ncbi_tax_id) REFERENCES "NcbiTaxs" (ncbi_tax_id),
        FOREIGN KEY(uniprot_id) REFERENCES "Uniprots" (uniprot_id)
);
CREATE INDEX "ix_Genbanks_genbank_accession" ON "Genbanks" (genbank_accession);
CREATE TABLE IF NOT EXISTS "Genbanks_Genomes" (
        genbank_id INTEGER NOT NULL,
        genome_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, genome_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(genome_id) REFERENCES "Genomes" (genome_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_CazyFamilies" (
        genbank_id INTEGER NOT NULL,
        family_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, family_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(family_id) REFERENCES "CazyFamilies" (family_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_Ecs" (
        genbank_id INTEGER NOT NULL,
        ec_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, ec_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
```

(continues on next page)

(continued from previous page)

```
        FOREIGN KEY(ec_id) REFERENCES "Ecs" (ec_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_Pdbs" (
        genbank_id INTEGER NOT NULL,
        pdb_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, pdb_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(pdb_id) REFERENCES "Pdbs" (pdb_id)
);
```

8.6 Retrieving structure files from PDB

The schema of a local CAZyme database can be retrieved using `cazy_webscraper`:

Alternatively, *sqlite3* can be used to retrieve the schema:

A visual representation of the db schema when using *cazy_webscraper* version >= 2.3.0 can be found here.

As of *cazy_webscraper* version >= 2.3.0, the schema of a local CAZyme database will be:

```
CREATE TABLE IF NOT EXISTS "Kingdoms" (
        kingdom_id INTEGER NOT NULL,
        kingdom VARCHAR,
        PRIMARY KEY (kingdom_id),
        UNIQUE (kingdom)
);
CREATE TABLE IF NOT EXISTS "GtdbTaxs" (
        gtdb_tax_id INTEGER NOT NULL,
        kingdom VARCHAR,
        phylum VARCHAR,
        tax_class VARCHAR,
        tax_order VARCHAR,
        family VARCHAR,
        genus VARCHAR,
        species VARCHAR,
        release VARCHAR,
        PRIMARY KEY (gtdb_tax_id),
        UNIQUE (kingdom, phylum, tax_class, tax_order, family, genus, species, release)
);
CREATE TABLE IF NOT EXISTS "CazyFamilies" (
        family_id INTEGER NOT NULL,
        family VARCHAR NOT NULL,
        subfamily VARCHAR,
        PRIMARY KEY (family_id),
        UNIQUE (family, subfamily)
);
CREATE INDEX fam_index ON "CazyFamilies" (family, subfamily);
CREATE TABLE IF NOT EXISTS "NcbiTaxs" (
        ncbi_tax_id INTEGER NOT NULL,
        kingdom VARCHAR,
        phylum VARCHAR,
```

(continues on next page)

(continued from previous page)

```
        tax_class VARCHAR,
        tax_order VARCHAR,
        family VARCHAR,
        genus VARCHAR,
        species VARCHAR,
        strain VARCHAR,
        PRIMARY KEY (ncbi_tax_id),
        UNIQUE (ncbi_tax_id)
);
CREATE INDEX ncbi_index ON "NcbiTaxs" (ncbi_tax_id, genus, species);
CREATE TABLE IF NOT EXISTS "Uniprots" (
        uniprot_id INTEGER NOT NULL,
        uniprot_accession VARCHAR,
        uniprot_name VARCHAR,
        sequence VARCHAR,
        seq_update_date VARCHAR,
        PRIMARY KEY (uniprot_id),
        UNIQUE (uniprot_accession)
);
CREATE INDEX uniprot_option ON "Uniprots" (uniprot_id, uniprot_accession);
CREATE TABLE IF NOT EXISTS "Ecs" (
        ec_id INTEGER NOT NULL,
        ec_number VARCHAR,
        PRIMARY KEY (ec_id),
        UNIQUE (ec_number)
);
CREATE INDEX "ix_Ecs_ec_number" ON "Ecs" (ec_number);
CREATE TABLE IF NOT EXISTS "Pdbs" (
        pdb_id INTEGER NOT NULL,
        pdb_accession VARCHAR,
        PRIMARY KEY (pdb_id),
        UNIQUE (pdb_accession)
);
CREATE INDEX pdb_idx ON "Pdbs" (pdb_accession);
CREATE TABLE IF NOT EXISTS "Logs" (
        log_id INTEGER NOT NULL,
        date VARCHAR,
        time VARCHAR,
        "database" VARCHAR,
        retrieved_annotations VARCHAR,
        classes VARCHAR,
        families VARCHAR,
        kingdoms VARCHAR,
        genera_filter VARCHAR,
        species_filter VARCHAR,
        strains_filter VARCHAR,
        ec_filter VARCHAR,
        cmd_line VARCHAR,
        PRIMARY KEY (log_id)
);
CREATE TABLE IF NOT EXISTS "Taxs" (
        taxonomy_id INTEGER NOT NULL,
```

(continues on next page)

(continued from previous page)

```
        genus VARCHAR,
        species VARCHAR,
        kingdom_id INTEGER,
        PRIMARY KEY (taxonomy_id),
        UNIQUE (genus, species),
        FOREIGN KEY(kingdom_id) REFERENCES "Kingdoms" (kingdom_id)
);
CREATE INDEX organism_option ON "Taxs" (taxonomy_id, genus, species);
CREATE TABLE IF NOT EXISTS "Genomes" (
        genome_id INTEGER NOT NULL,
        assembly_name VARCHAR,
        gbk_version_accession VARCHAR,
        gbk_ncbi_id INTEGER,
        refseq_version_accession VARCHAR,
        refseq_ncbi_id INTEGER,
        gtdb_tax_id INTEGER,
        PRIMARY KEY (genome_id),
        UNIQUE (assembly_name, gbk_version_accession, refseq_version_accession),
        FOREIGN KEY(gtdb_tax_id) REFERENCES "GtdbTaxs" (gtdb_tax_id)
);
CREATE INDEX genome_options ON "Genomes" (assembly_name, gbk_version_accession, refseq_
↪version_accession);
CREATE TABLE IF NOT EXISTS "Genbanks" (
        genbank_id INTEGER NOT NULL,
        genbank_accession VARCHAR,
        sequence VARCHAR,
        seq_update_date VARCHAR,
        taxonomy_id INTEGER,
        ncbi_tax_id INTEGER,
        uniprot_id INTEGER,
        PRIMARY KEY (genbank_id),
        UNIQUE (genbank_accession),
        FOREIGN KEY(taxonomy_id) REFERENCES "Taxs" (taxonomy_id),
        FOREIGN KEY(ncbi_tax_id) REFERENCES "NcbiTaxs" (ncbi_tax_id),
        FOREIGN KEY(uniprot_id) REFERENCES "Uniprots" (uniprot_id)
);
CREATE INDEX "ix_Genbanks_genbank_accession" ON "Genbanks" (genbank_accession);
CREATE TABLE IF NOT EXISTS "Genbanks_Genomes" (
        genbank_id INTEGER NOT NULL,
        genome_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, genome_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(genome_id) REFERENCES "Genomes" (genome_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_CazyFamilies" (
        genbank_id INTEGER NOT NULL,
        family_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, family_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(family_id) REFERENCES "CazyFamilies" (family_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_Ecs" (
```

(continues on next page)

(continued from previous page)

```
        genbank_id INTEGER NOT NULL,
        ec_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, ec_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(ec_id) REFERENCES "Ecs" (ec_id)
);
CREATE TABLE IF NOT EXISTS "Genbanks_Pdbs" (
        genbank_id INTEGER NOT NULL,
        pdb_id INTEGER NOT NULL,
        PRIMARY KEY (genbank_id, pdb_id),
        FOREIGN KEY(genbank_id) REFERENCES "Genbanks" (genbank_id),
        FOREIGN KEY(pdb_id) REFERENCES "Pdbs" (pdb_id)
);
```

8.7 Retrieving data from UniProt

`cazy_webscraper` can be used to retrieve user-specified data sets from the UniProt database, for a given subset of proteins in a local CAZyme database created using `cazy_webscraper`. The `cazy_webscraper` application can be invoked *via* the command line

8.7.1 Quick Start

To download UniProt protein accessions and names from UniProt for all protein in the local CAZyme database, and save the data to the local CAZyme database, use the following command structure:

```
cw_get_uniprot_data <path to local CAZyme db>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

Warning: Please do not download data from UniProt for the entire CAZy database unless absolute necessary. Retrieving the data from any of these exteranl databases for the entire CAZy dataset will take several hours and may unintentionally deny the service to others.

To download UniProt protein accessions and names from UniProt for all protein in the local CAZyme database, including EC number annotations, PDB accessions and protein sequences, and save the data to the local CAZyme database, use the following command structure:

```
cw_get_uniprot_data <path to local CAZyme db> --ec --pdb --sequence
```

For example:

8.7.2 Command line options

Below are listed the required and optional command-line options for configuring the retrieval of data from UniProt.

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec`, `-e` - Enable retrieval of EC number annotations from UniProt. Default, EC number annotations are **not** retrieved.

`--ec_filter` - List of EC numbers to limit the retrieval of protein data for proteins annotated with at least one of the given EC numbers **in the local CAZyme database**.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--force` - Force writing in existing cache directory.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomic kingdoms to restrict the scrape to. Default: None, filter is not applied.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--name_update` - Boolean, whether to overwrite the existing protein name (previously retrieved from UniProt). Default: do not update.

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_log` - When called, content in the existing log dir will **not** be deleted. Default: False (existing content is deleted).

`--pdb`, `-p` - Enable retrieval of PDB accessions. Default, PDB accessions not retrieved.

`--retries`, `-r` - Define the number of times to retry making a connection to CAZy if the connection should fail. Default: 10.

`--sequence`, `-s` - Enable retrieving protein amino acid sequences. Default, sequences are **not** retrieved.

`--update_name` - If a newer version of the protein name is available, overwrite the existing name for the protein in the database. Default is false, the protein name is **not** overwritten and updated.

`--update_seq` - If a newer version of the protein sequence is available, overwrite the existing sequence for the protein in the database. Default is false, the protein sequence is **not** overwritten and updated.

`--delete_old_ec_relationships` - Boolean, delete old Genbanks-EC number relationships - For those proteins in the local db for whom data is downloaded from UniProt, compare the current links between the proteins in the Genbanks table and EC numbers in the Ecs table. Delete Genbanks-Ecs relationships that are not longer listed in the respective protein records in UniProt.

`--delete_old_ecs` - Boolean, delete EC number - Delete EC numbers that are not linked to any proteins listed in the Genbanks table. These can arise from multiple retrievals of data from the UniProt data over a period of time during UniProt records have been updated.

`--delete_old_pdb_relationships` - Boolean, delete old Genbanks-PDB relationships - For those proteins in the local db for whom data is downloaded from UniProt, compare the current links between the proteins in the Genbanks table and PDB accessions in the Pdbs table. Delete Genbanks-Pdbs relationships that are not longer listed in the respective protein records in UniProt.

`--delete_old_pdbs` - Boolean, delete PDB accessions - Protein relationships that are no longer listed in UniProt, i.e. an PDB accessions that are no longer included in UniProt but is in the local database. If set to TRUE these relationships will be DELETED from the database.

`--use_uniprot_cache` - Path to a JSON file, keyed by UniProt accessions/IDs and valued by dicts containing *{'gbk_acc': str, 'db_id': int}*. This file part of the cache created by *cw_get_uniprot_data*. This is option to skip retrieving the UniProt IDs for a set of GenBank accessions, if retrieving data for the same dataset (this save a lot of time!)

`skip_download` - Bool, default False. If set to True, only uses data from UniProt cache and will not download new data from UniProt.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--taxonomy`, `-t` - Retrieve taxonomic classifications (genus species) and add to the local CAZyme db.

`--timeout` - Connection timout limit (seconds). Default: 45.

`--use_uniprot_cache` - Path to JSON file containing data previosuly retrieved from UniProt by `cazy_webscraper`, use if an error occurred while adding the data to the local CAZyme database. This will skip the retrieval of data from UniProt, and the cached data will be added to the local CAZyme database. This can also be shared with others to add the same data to their local CAZyme database.

`--bioservices_batch_size` - Size of an individual batch query submitted to the *UniProt REST API <https://www.uniprot.org/help/programmatic_access>_* to retrieve protein data from UniProt. Default is 1000.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.7.3 Batch sizes

Note that according to Uniprot (June 2022), there are various limits on ID Mapping Job Submission:

Limit	Details
100,000	Total number of ids allowed in comma separated param ids in /idmapping/run api
500,000	Total number of "mapped to" ids allowed
100,000	Total number of "mapped to" ids allowed to be enriched by UniProt data
10,000	Total number of "mapped to" ids allowed with filtering

8.7.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of protein data from UniProt. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which data will be retrieved from UniProt, others (e.g. `--species`) are used to filter and fine-tune the protein dataset for which protein data will be retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy as retrieving data from UniProt. Examples of using these flags can be found in the `cazy_webscraper` tutorial in this documentation.

Here we discuss using the new flags `--ec`, `--pdb`, `--sequence`, `--seq_update`, and `--ec_filter`.

Note: To retrieve data for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

Note: The command for retrieving protein data from UniProt for proteins in a local CAZyme database is `cw_get_uniprot_data`.

8.7.5 Data retrievable from UniProt

By default `cw_get_uniprot_data` retrieves the UniProt protein accession and protein name from UniProt, for proteins in a local CAZyme database. `cw_get_uniprot_data` can also retrieve from UniProt:

- EC number annotations
- PDB accessions
- Protein amino acid sequences

Warning: When performing retrievals of large datasets from UniProt, please perform these retrievals during quiet periods (e.g. at the weekend).

Warning: It is strongly advised to only download data from UniProt for the necessary protein datasets. Retrieval of very large datasets from UniProt (e.g. for +1,000,000 proteins or all proteins in the GH class) can result in UniProt terminating the connection early due to a high bandwidth command over an extended period of time.

To retrieve data from large datasets, it is recommend to break down the dataset into subgroups, and periodically retrieve the data for one subgroup. This reduces the burdeon on the UniProt server and will not break the expected use and practises of the UniProt database.

Retrieving EC number annotations

To retrieve EC number annotations from UniProt add the `--ec` / `-e` flag to the command:

```
cw_get_uniprot_data cazy_db.db --ec
```

OR

```
cw_get_uniprot_data cazy_db.db -e
```

Note: **All** EC number annotations are retrieved for all CAZymes matching the given filter criteria. In the example command above, no filters were provided therefore, all EC number annotations will be retrieved for all CAZymes in the local CAZyme database (in this case called `cazy_db.db`).

Retrieving PDB accessions

To retrieve all PDB accessions for all CAZymes in the local CAZyme database matching the given filter criteria, add the `--pdb` / `-p` flag to the command:

```
cw_get_uniprot_data cazy_db.db --pdb
```

OR

```
cw_get_uniprot_data cazy_db.db -p
```

Retrieving protein sequences

To retrieve all protein amino acid sequences for all CAZymes in the local CAZyme database matching the given filter criteria, add the `--sequence` / `-s` flag to the command:

```
cw_get_uniprot_data cazy_db.db --sequence
```

OR

```
cw_get_uniprot_data cazy_db.db -s
```

`cw_get_uniprot_data` stores the protein amino acids sequence within the local CAZyme database, as well as the 'last modified date' retrieved from UniProt.

Updating local sequences

When using `--sequence` flag, `cw_get_uniprot_data` will only add *new* protein sequences to the database, i.e. it will only add protein sequences to records that do not have a sequence. Therefore, if a protein already has a sequence in the local database, this sequence is **not** overwritten.

You may wish to update the protein sequences in your local CAZyme database. To do this use the `--sequence` / `-s` flag to tell `cw_get_uniprot_data` to retrieve protein sequences, **and** use the `--seq_update` flag.

```
cw_get_uniprot_data cazy_db.db -s --seq_update
```

This instructs `cw_get_uniprot_data` to overwriting existing protein sequences in the local database *if* a newer version of the sequence is retrieved from UniProt. This is checked by comparing the 'last modified date' of the protein sequence in the local database against the sequence retrieved from UniProt.

8.7.6 Using the EC number filter

Having previously retrieved EC number annotations and added them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cw_get_uniprot_data` will retrieve the specified UniProt data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** only CAZymes will all provided EC numbers will have data retrieved from UniProt for them.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cw_get_uniprot_data` to retrieve data for protein A. This is because `cw_get_uniprot_data` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cw_get_uniprot_data` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.7.7 Configuration using a YAML file

As with scraping CAZy, a YAML file can be provided to define the filters for retrieving data from UniProt. The same YAML file can be used both for scraping CAZy and UniProt. However, the configuration file for retrieving data from UniProt can contain the additionl `ec` tag.

Using a config file supports reproducible documentation of `cazy_webscraper` usage.

An template YAML file is provided in the `cazy_webscraper` repository (`configuration_files/template-get_data_config.yaml`):

```
# Under 'classes' list class from which all proteins will retrieved
# Under each families respective name, list the specific families/subfamilies to be␣
↪scraped
```

(continues on next page)

(continued from previous page)

```
# Write the FULL family name, e.g. 'GH1', NOT only its number, e.g. '1'
# To list multiple families, each familiy must be on a new line starting indented once
# relative to the parent class name, and the name written within quotation marks.
# For more information on writing lists in Yaml please see:
# https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
classes:  # classes from which all proteins will be retrieved
- "GH"
- "CE"
Glycoside Hydrolases (GHs):
GlycosylTransferases (GTs):
Polysaccharide Lyases (PLs):
- "GT1"
- "GT5"
- "GT6"
Carbohydrate Esterases (CEs):
Auxiliary Activities (AAs):
Carbohydrate-Binding Modules (CBMs):
genera:  # list genera to be scraped
- "Trichoderma"
- "Aspergillus"
species:  # list species, this will scrape all strains under the species
- "Pythium ultimum"
strains:  # list specific strains to be scraped
kingdoms:  # Archaea, Bacteria, Eukaryota, Viruses, Unclassified
ec:
- "EC1.2.3.4"
```

Attention: The YAML configuration file must contain all tags/headings indicated in the example configuration file found in the repository:

- classes
- Glycoside Hydrolases (GHs)
- GlycosylTransferases (GTs)
- Polysaccharide Lyases (PLs)
- Carbohydrate Esterases (CEs)
- Auxiliary Activities (AAs)
- Carbohydrate-Binding Modules (CBMs)
- genera
- species
- strains
- kingdoms
- ec

Each value in the YAML mappings for these arguments must be listed on a separate line, indented by 4 spaces, and the class name encapsulated with single or double quotation marks. For example:

```
classes:
    - "GT"
    - "pl"
Glycoside Hydrolases (GHs):
    - "GH1"
    - "GH2"
```

Synonyms for CAZy classes

A number of synonyms may be provided for CAZy classes, e.g. both "GH" and "Glycoside-Hydrolases" are accepted as synonyms for "Glycoside Hydrolases (GHs)" (the name recorded at CAZy). These alternatives are defined in the `cazy_webscraper` repository, in the file `scraper/utilities/parse_configuration/cazy_dictionary.json`.

8.8 Tutorials on configuring `cazy_webscraper` to retrieve data from UniProt

`cazy_webscraper` can be configured to retrieve user specified data sets from UniProt for specific date sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

Note: `cazy_webscraper` retrieves protein data from UniProt for CAZymes in a local CAZyme database. It will **not** add new proteins to the database.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of data from UniProt. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_get_uniprot_data` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/uniprot/get_uniprot_data.py`.

From this point on, we will be discusseing the `cw_get_uniprot_data` command, which is used by `cazy_webscraper` for retrieving data from UniProt. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

All data retrieved from UniProt by `cw_get_uniprot_data` is added to the local CAZyme database.

8.8.1 Configuration via the command line

`cw_get_uniprot_data` only requires one argument: the path to the local CAZyme database created using `cazy_webscraper`.

Therefore, `cw_get_uniprot_data` can be enabled using a simple command structure:

```
cazy_webscraper <path to the local CAZyme db>
```

For example, if our database was stored in `cazy/cazyme.db`, we would used:

```
cazy_webscraper cazy/cazyme.db
```

Note: Make sure `cw_get_uniprot_data` is pointed directly at the database file.

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to:

- Retrieve protein data for **all** CAZymes in the local CAZyme db
- Retrieve UniProt protein accessions and protein names
- Not retrieve EC number, PDB accessions and protein sequences

8.8.2 Options configurable at the command line

The following behaviours of the `cw_get_uniprot_data` can be configured at the command-line in the terminal:

- Limit the retrieve of UniProt protein data to CAZymes in the local databaes from specific CAZy classes, CAZy families, kingdoms, genuera, species, strains and/or EC numbers
- Enable retrieving EC number annotations
- Enable retrieving PDB accessions
- Enable retrieving protein amino acid sequences
- Enable updating protein sequences in the local CAZyme database if newer versions are retrieved from UniProt
- Enable verbose logging during the operation of the webscraper

Here you can find a full list of the command-line flags and options.

8.8.3 Retrieving protein data for CAZy classes and families to scrape

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieve data from UniProt.

For instance, if instead of retrieving protein data for all CAZymes in your local CAZyme database, you want to retrieve protein data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_uniprot_data cazy/cazyme.db --classes GH,CE
```

OR

```
cw_get_uniprot_data cazy/cazyme.db --classes Glycoside Hydrolases,Carbohydrate Esterases
```

Retrieving protein data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein data for all proteins in PL1, PL2 and PL3 in the local CAZyme database use the following command:

```
cw_get_uniprot_data cazy/cazyme.db --families PL1,PL2,PL3
```

Warning: `cw_get_uniprot_data` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_uniprot_data cazy/cazyme.db --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cw_get_uniprot_data cazy/cazyme.db --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

8.8.4 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have protein data retrieved from UniProt and added to the local CAZyme datbase.

For example, if you want to retrieve protein data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species then use the command

```
cw_get_uniprot_data cazy/cazyme.db --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve protein data for all CAZymes in a local CAZyme database that are derived from all viral

species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_uniprot_data` command. The command we would use would be:

```
cw_get_uniprot_data cazy/cazyme.db --kingdoms viruses --genera Aspergillus --species␣
↪Layia carnosa,Layia chrysanthemoides --strains Trichoderma reesei QM6a,Trichoderma␣
↪reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_uniprot_data` will retrieval CAZymes from *all* strains of the species.

8.8.5 Applying EC number filter

The retrieval of protein data from UniProt can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations and added them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_uniprot_data cazy/cazyme.db --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cazy_webscraper` will retrieve the specified UniProt data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** only CAZymes will all provided EC numbers will have data retrieved from UniProt for them.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cazy_webscraper` to retrieve data for protein A. This is because `cazy_webscraper` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cazy_webscraper` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.8.6 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom protein data from UniProt will be retrieved. These flags can be used with any combination of `--ec`, `--pdb`, `--sequence`, `--update_seq` to customise what data is retrieved from UniProt and added to the local CAZyme database.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To retrieve PDB accessions for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species we use the command:

```
cw_get_uniprot_data cazy/cazyme.db --pdb --classes GH,CE --families CE1,CE5,CE8 --
↪kingdoms bacteria
```

Example 2: To retrieve EC numbers, PDB accessions and taxonomies for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species we use the command:

```
cw_get_uniprot_data cazy/cazyme.db --pdb --ec --classes GH --genera Aspegillus,
↪Trichoderma --taxonomy
```

Example 3: To retrieve EC numbers and sequences for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_uniprot_data cazy/cazyme.db --ec --sequences --classes GH,CE,CBM --kingdoms␣
↪bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

8.8.7 Providing a list of accessions

Instead of retrieving protein data for all CAZymes matching a defined set of criteria, `cw_get_uniprot_data` can retrieve protein data a set of CAZymes defined by their GenBank accession.

The flag `--genbank_accessions` can be used to provide `cw_get_uniprot_data` a list of GenBank accessions to identify the specific set of CAZymes to retrieve protein data for.

The list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_uniprot_data` via the `--genbank_accessions` flag.

Warning: `--genbank_accessions` takes president over the filter flags.

When `--genbank_accessions` is used, `cw_get_uniprot_data` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_uniprot_data` will ignore the `--classes` flag and only retrieve protein data for the proteins listed in the file provided via the `--genbank_accessions`.

8.9 Retrieving Protein Sequences from GenBank

`cazy_webscraper` can be used to retrieve protein amino acid sequences from NCBI GenBank for user-specified data sets of CAZymes in the local CAZymes database.

The retrieval of data from NCBI is performed by using the **BioPython** *Bio.entrez <https://biopython.org/docs/1.75/api/Bio.Entrez.html>_* module [Cock *et al.*, 2009].

> Cock, P. J. A, Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A. _et al._ (2009) 'Biopython: freely available Python tools for computaitonal molecular biology and bioinformatics', _Bioinformatics_, 25(11), pp. 1422-3.

Note: For specific information of the `Bio.entrez` module please see the entrez documentation.

8.9.1 Quick Start

To download protein sequences for all CAZymes in the local CAZyme database, and write them to the local CAZyme database, use the following command structure:

```
cw_get_genbank_seqs 'path to local CAZyme db' 'user email address'
```

For example:

```
cw_get_genbank_seqs cazy/cazyme.db myemail@domain.com
```

Note: The `cw` prefix is an abbreviation of `cazy_webscraper`.

8.9.2 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`email` - **REQUIRED** User email address, required by NCBI Entrez.

`--batch_size` - Size of batch query posted to NCBI Entrez. Default 150.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_data` - Path to a txt file downloaded from CAZy containing a CAZy database dump

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec_filter` - List of EC numbers to limit the retrieval of protein data for proteins annotated with at least one of the given EC numbers **in the local CAZyme database**.

`--force`, `-f` - Force writing cachce to exiting cache directory.

`--file_only`, `-F` - Only add seqs provided via JSON and/or FASTA file. Do not retrieved data from NCBI.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_log` - When called, content in the existing log dir will **not** be deleted. Default: False (existing content is deleted).

`--retries`, `-r` - Define the number of times to retry making a connection to CAZy if the connection should fail. Default: 10.

`--seq_dict`, - Path to a JSON file, keyed by GenBank accessions and valued by protein sequence. This file is created as part of the cache, after all protein sequences are retrieved from GenBank. This skips the retrieval of the protein sequences from GenBank only for those seqs included in the file.

`--seq_file`, - Path to a JSON file, keyed by GenBank accessions and valued by protein sequence. This file is created as part of the cache, after all protein sequences are retrieved from GenBank. This skips the retrieval of the protein sequences from GenBank only for those seqs included in the file.

`--seq_update` - If a newer version of the protein sequence is available, overwrite the existing sequence for the protein in the database. Default is false, the protein sequence is **not** overwritten and updated.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.9.3 Basic Usage

The command-line options listed above can be used in combination to customise the scraping of CAZy. Some options (e.g. `--families` and `--classes`) define the broad group of data that will be scraped, others (e.g. `--species`) are used to filter and fine-tune the data that is scraped.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy as retrieving protein sequences from GenBank and protein data from UniProt. Examples of using these flags can be found in the tutorial.

The `--seq_update` flag is used in the same way for retrieving protein sequences from UniProt and GenBank.

Note: To retrieve data for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.9.4 Updating local sequences

When using `--sequence` flag, `cazy_webscraper` will only add *new* protein sequences to the database, i.e. it will only add protein sequences to records that do not have a sequence. Therefore, if a protein already has a sequence in the local database, this sequence is **not** overwritten.

You may wish to update the protein sequences in your local CAZyme database. To do this use the `--sequence`/`-s` flag to tell `cazy_webscraper` to retrieve protein sequences, **and** use the `--seq_update` flag.

```
cw_get_genbank_seqs cazy_db.db -s --seq_update
```

This instructs `cazy_webscraper` to overwriting existing protein sequences in the local database *if* a newer version of the sequence is retrieved from UniProt. This is checked by comparing the 'last modified date' of the protein sequence in the local database against the sequence retrieved from UniProt.

8.10 Retrieving Sequences from GenBank Tutorial

`cazy_webscraper` can be configured to retrieve protein sequences for for specific date sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of protein sequences from CAZy, UniProt, GenBank and PDB.

Note: `cazy_webscraper` retrieves protein sequences from GenBank for CAZymes in a local CAZyme database.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of protein sequences from GenBank. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_get_genbank_seqs` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/uniprot/get_uniprot_data.py`.

From this point on, we will be discusseing the `cw_get_genbank_seqs` command, which is used by `cazy_webscraper` for retrieving protein sequences from GenBank. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

All protein sequences from GenBank by `cw_get_genbank_seqs` are added to the local CAZyme database.

8.10.1 Configuration via the command line

`cw_get_genbank_seqs` requires two arguments: * The path to the local CAZyme database created using `cazy_webscraper` * The user's email address

Therefore, `cw_get_genbank_seqs` can be enabled using a simple command structure:

```
cazy_webscraper <path to the local CAZyme db> <email address>
```

Note: NCBI Entrez is used to retrieve the data from GenBank and requires the user's email address.

For example, if our database was stored in `cazy/cazyme.db`, we would used:

```
cazy_webscraper cazy/cazyme.db my_email@domain.com
```

Note: Make sure `cw_get_genbank_seqs` is pointed directly at the database file.

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: retrieve protein sequences for **all** CAZymes in the local CAZyme db

8.10.2 Options configurable at the command line

The following behaviours of the `cw_get_genbank_seqs` can be configured at the command-line in the terminal:

- Limit the retrieval of protein sequences to CAZymes in the local databaes from specific CAZy classes, CAZy families, kingdoms, genuera, species, strains and/or EC numbers
- Enable updating protein sequences in the local CAZyme database if newer versions are retrieved from UniProt
- Enable verbose logging during the operation of the webscraper

Here you can find a full list of the command-line flags and options.

8.10.3 Retrieving protein sequences for CAZy classes and families to scrape

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving protein sequences from GenBanks.

For instance, if instead of retrieving protein sequences for all CAZymes in your local CAZyme database, you want to retrieve protein sequences for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein sequences for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein sequences for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --classes GH,CE
```

OR

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --classes Glycoside␣
↪Hydrolases,Carbohydrate Esterases
```

Retrieving protein sequences for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein sequences for all proteins in PL1, PL2 and PL3 in the local CAZyme database use the following command:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --families PL1,PL2,PL3
```

Warning: `cw_get_genbank_seqs` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein sequences for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --families PL1,PL2,PL3 --
↪classes GH,CE
```

AND

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --classes GH,CE --families␣
↪PL1,PL2,PL3
```

are accepted.

8.10.4 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein sequences by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have protein sequences retrieved from GenBank and added to the local CAZyme datbase.

For example, if you want to retrieve protein sequences for all CAZymes in a local CAZyme database from bacterial and eukaryotic species then use the command

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve protein sequences for all CAZymes in a local CAZyme database that are derived from all viral

species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_genbank_seqs` command. The command we would use would be:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --kingdoms viruses --genera␣
↪Aspergillus --species Layia carnosa,Layia chrysanthemoides --strains Trichoderma␣
↪reesei QM6a,Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_genbank_seqs` will retrieval CAZymes from *all* strains of the species.

8.10.5 Applying EC number filter

The retrieval of protein sequences from GenBank can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations and added them to the local CAZyme database, you may wish to retrieve protein sequences for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --ec_filter "EC1.2.3.4,EC2.3.
↪4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cazy_webscraper` will retrieve the specified UniProt data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** only CAZymes will all provided EC numbers will have data retrieved from UniProt for them.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cazy_webscraper` to retrieve data for protein A. This is because `cazy_webscraper` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cazy_webscraper` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.10.6 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom protein sequences from GenBank will be retrieved.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To retrieve protein sequences for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species we use the command:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --classes GH,CE --families␣
↪CE1,CE5,CE8 --kingdoms bacteria
```

Example 2: To protein sequences for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species we use the command:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk -classes GH --genera␣
↪Aspegillus,Trichoderma
```

Example 3: To retrieve protein sequences for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_genbank_seqs cazy/cazyme.db dummy.email@domain.co.uk --ec --sequences --classes␣
↪GH,CE,CBM --kingdoms bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

8.10.7 Providing a list of accessions

Instead of retrieving protein sequences for all CAZymes matching a defined set of criteria, `cw_get_genbank_seqs` can retrieve protein sequences a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_get_genbank_seqs` a list of GenBank accessions to identify the specific set of CAZymes to retrieve protein sequences for.

The flag `--uniprot_accessions` can be used to provide `cw_get_genbank_seqs` a list of UniProt accessions to identify the specific set of CAZymes to retrieve protein sequences for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_genbank_seqs` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_get_genbank_seqs` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_genbank_seqs` will ignore the `--classes` flag and only retrieve protein sequences for the proteins listed in the file provided via the `--genbank_accessions`.

8.10.8 Providing sequences from a file

While `cw_get_genbank_seqs` is retrieving protein sequences from NCBI, the retrieved protein sequences are written to a FASTA file in the cache directory.

To add sequences from a cached FASTA file (e.g. to continue a download that was previously interrupted) and/or add GenBank sequences from a previous download (e.g. by a colleage), use the `--seq_file` flag followed by the path to the FASTA containing the protein sequences to be added to the database. The ID for each sequence **must** be the NCBI protein version accession.

`cw_get_genbank_seqs` also generates a JSON file of the cached sequences. To add sequences from the cached JSON file to the local CAZyme database, use the `--seq_dict` flag followed by the path to the JSON file.

By default `cw_get_genbank_seqs` will add sequences retrieved from the FASTA and/or JSON file **and** will retrieve protein sequences from NCBI for proteins matching the provided criteria to define proteins of interest.

To add **only** the sequences from a FASTA and/or JSON file, and **not** download any sequences from NCBI, use the `--file_only` flag.

8.11 Extract protein squences from the local database

`cazy_webscraper` can be used to extract protein sequences stored in the local CAZyme database.

The extracted protein sequences can be written to any combination of: * A single FASTA file containing *all* extracted sequences * One FASTA file per extracted sequence * A BLASTp database

GenBank and/or UniProt protein sequences retrieved can be extracted.

8.11.1 Quick Start

To extract all protein sequences previously retrieved from GenBank and UniProt, use the following command structure:

```
cw_extract_db_seqs <path to local CAZyme db> genbank uniprot
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

Note: 'genbank' and 'uniprot' are not case sensitive, therefore, both GenBank and UniProt are also accepted.

Warning: At least one database (either GenBank or UniProt) **must** be provided.

8.11.2 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`source` - **REQUIRED** Define source databases of protein sequences. Accepts 'genbank' and 'uniprot'. To list both, separate with a single space, e.g.

```

```

`cw_extract_sequence cazy_database.db genbank uniprot

The database names are not case sensitive, therefore, both GenBank and genbank are accepted.

`--blastdb`, `-b` - Create BLAST database of extracted protein sequences. Provide the path to the directory to store the BLAST database in.

`--fasta_dir` - Write out each extracted sequence to a separate FASTA file in the provided dir. Provide a path to a directory to write out the FASTA files.

`--fasta_file` - Write out all extracted sequences to a single FASTA file. Provide a path to write out the FASTA file.

Warning: At least one of `--blastdb`, `--fasta_dir`, and `--fasta_file` must be called to inform *cazy_webscraper* where to write the output to. If none are called sequences will be extracted._

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - List of classes from which all families are to be scrape.

`--ec_filter` - Limist retrieval of protein data to proteins annotated with a provided list of EC numbers. Separate the EC numbers bu single commas without spaces. Recommend to wrap the entire str in quotation marks, for example: .. code-block:: bash

    cw_get_uniprot_data my_cazyme_db/cazyme_db.db –ec_filter 'EC1.2.3.4,EC2.3.1.-'

`--force`, `-f` - Force overwriting exsiting files and writing to existing output directory.

`--families` - List of CAZy (sub)families to scrape.#

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--genbank_accession` - Path to text file containing a list of GenBanks accessions to extract protein sequences for. A unique accession per line.

`--genera` - List of genera to restrict the scrape to. Default: None, filter not applied to scrape.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete` - When called, content in the existing output dir will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species written as Genus Species) to restrict the scraping of CAZymes to. CAZymes will be retrieved for **all** strains of each given species.

`--strains` - List of specific species strains to restrict the scraping of CAZymes to.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to extract protein sequences for. A unique accession per line.

`--verbose`, `-v` - Enable verbose logging. This does not set the SQLite engine *echo* parameter to True. Default: False.

8.11.3 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval the extraction of protein sequences to proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins, others (e.g. `--species`) are used to filter and fine-tune the protein dataset.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy and UniProt. Examples of using these flags can be found in the `cazy_webscraper` and `cw_get_uniprot_data` tutorial in this documentation.

Note: To extract protein sequences for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.12 Tutorials on configuring the extraction of protein sequences

`cazy_webscraper` can be configured to extract GenBank and/or UniProt protein sequences for user specified sets of proteins from a local CAZyme database. Many of the configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the extraction of protein sequences. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_extract_db_seqs` - this is the method used in this tutorial. If you installed `cazy_webscraper`

from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/extract/extract_sequences.py`.

From this point on, we will be discussed the `cw_extract_db_seqs`, which is the entry point for extract protein sequences from the local CAZyme database. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

8.12.1 Configuration via the command line

`cw_extract_db_seqs` has at 2 required arguments:

1. The path to the local CAZyme databases created using `cazy_webscraper`
2. The names of the database from which the proteins were sourced

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: Extract (GenBank and/ot UniProt) protein sequences for **all** CAZymes in the local CAZyme db

Defining the source of the proteins

Protein sequences previously retrieved from GenBank **and/or** UniProt can be extracted.

- To extract only protein sequences from GenBank, use `genbank`.
- To extract only protein sequences from UniProt, use `uniprot`.
- To extract GenBank and UniProt protein sequences, use `genbank uniprot` or `uniprot genbank`.

Tip: The order the databases (i.e. 'genbank' and 'uniprot') does not matter, and they are **not** casesensitive.

`cw_extract_db_seqs` can be enabled using a simple command structure:

```
cw_extract_db_seqs <path to the local CAZyme db> genbank uniprot
```

For example, if our database was stored in `cazy/cazyme.db`, we would used:

```
cw_extract_db_seqs cazy/cazyme.db genbank uniprot
```

Defining the target output

The extracted protein sequences can be written to any combination of the following: * A single FASTA file containing all extracted protein sequences * One FASTA file per extracted sequence * A BLAST database

To define a **single FASTA file** to write all extracted sequences to use the `--fasta_file` flag, followed by the path to the target file. These file nor its parent directorties need to already exist, `cazy_webscraper` will build all necessary parent directories. For example:

```
cw_extract_db_seqs cazy/cazyme_database.db genbank --fasta_file cazy/protein_sequences/
↪all_genbanks.FASTA
```

To write out each extracted sequence **to its own FASTA file** use the `--fasta_dir` flag, followed by the path to the target directory. `cazy_webscraper` will build all necessary parent directories. For example:

```
cw_extract_db_seqs cazy/cazyme_database.db genbank --fasta_dir cazy/protein_sequences
```

To build a BLAST database of the extracted protein squences, use the `--blastdb` or `-b` flag, followed by the path of where to write out the database, including the database name. `cazy_webscraper` will build all necessary parent directories. For example:

```
cw_extract_db_seqs cazy/cazyme_database.db genbank --blastdb cazy/protein_sequences/all_
↪genbanks_blast_db.db
```

Any combination of `--fasta_file`, `--fasta_dir` and `--blastdb` can be used to produce multiple outputs. For example, to generate a single FASTA file of all extracted UniProt protein sequences **and** write the extracted sequences to a BLAST database:

```
cw_extract_db_seqs cazy/cazyme_database.db genbank \
    --fasta_file cazy/protein_sequences/all_genbanks.FASTA \
    --blastdb cazy/protein_sequences/all_genbanks_blast_db.db
```

Tip: Backward slashes '\' can be used to break up a long command into multiple lines to make it easier to read.

8.12.2 FASTA file formats

The FASTA files generated by `cw_extract_db_seqs` have a very simple protein ID line. The line always and only contains: * The GenBank or UniProt accession * The name of the source database: 'GenBank' or 'UniProt'

For example, a protein sequence from GenBank which is extracted from a local CAZyme datbase will be presented as:

8.12.3 Options configurable at the command line

The following behaviours of the `cw_extract_db_seqs` can be configured at the command-line in the terminal to limit the extraction of protein sequences to CAZymes in the local databaes from specific:

- CAZy classes
- CAZy families and subfamilies
- Taxonomic kingdoms
- Genuera
- Species
- Species strains
- Annotated with at least one of a set of specified EC numbers

Here you can find a full list of the command-line flags and options.

8.12.4 Extract protein sequences for specific CAZy classes and families

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for extracting protein sequences for proteins of interest.

For instance, if instead of extracting protein sequences for all CAZymes in your local CAZyme database, you want to extract protein sequebces for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to extract protein sequences for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to extract protein sequences for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_extract_db_seqs cazy/cazyme.db genbank --classes GH,CE
```

OR

```
cw_extract_db_seqs cazy/cazyme.db genbank --classes 'Glycoside Hydrolases','Carbohydrate␣
↪Esterases'
```

Warning: When including spaces in a parameter value, such as 'Glycoside Hydrolases' single or double quotation marks must be written around the value.

Extracting protein sequences for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to extract GenBank protein sequences for all proteins in PL1, PL2 and PL3 in the local CAZyme database use the following command:

```
cw_extract_db_seqs cazy/cazyme.db genbank --families PL1,PL2,PL3
```

Warning: `cw_extract_db_seqs` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To extract UniProt protein sequences for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_extract_db_seqs cazy/cazyme.db uniprot --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cw_extract_db_seqs cazy/cazyme.db uniprot --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

8.12.5 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to extract protein sequences by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have protein data retrieved from UniProt and added to the local CAZyme datbase.

For example, if you want to extract GenBank protein sequences for all CAZymes in a local CAZyme database from bacterial and eukaryotic species then use the command

```
cw_extract_db_seqs cazy/cazyme.db genbank --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to extract GenBank and UniProt protein sequences for all CAZymes in a local CAZyme database that are derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_extract_db_seqs` command. The command we would use would be:

```
cw_extract_db_seqs cazy/cazyme.db genbank uniprot --kingdoms viruses --genera␣
↪Aspergillus --species Layia carnosa,Layia chrysanthemoides --strains Trichoderma␣
↪reesei QM6a,Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_extract_db_seqs` will retrieval CAZymes from *all* strains of the species.

8.12.6 Applying EC number filter

The extraction of protein sequences an also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations from UniProt and added them to the local CAZyme database, you may wish to extract protein sequences for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_extract_db_seqs cazy/cazyme.db genbank --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cazy_webscraper` will retrieve the specified UniProt data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** only CAZymes will all provided EC numbers will have data retrieved from UniProt for them.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cazy_webscraper` to retrieve data for protein A. This is because `cazy_webscraper` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cazy_webscraper` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.12.7 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom protein sequences will be extracted. These flags can be used with any combination of `--ec`, `--pdb`, `--sequence`, `--update_seq` to customise what data is retrieved from UniProt and added to the local CAZyme database.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To extract GenBank protein sequences for CAZymes: * In GH, GT, CE1, CE5 and CE8 * Derived from bacterial species

```
cw_extract_db_seqs cazy/cazyme.db genbank --classes GH,CE --families CE1,CE5,CE8 --
↪kingdoms bacteria
```

Example 2: To extract GenBank protein sequences for CAZymes: * In GH * From *Aspegillus* and *Trichoderma* species .. code-block:: bash

    cw_extract_db_seqs cazy/cazyme.db genbank –classes GH –genera Aspegillus,Trichoderma

Example 3: To extract GenBank and UniProt protein sequences for CAZymes: * In GH,CE and CBM * Derived from baceterial species * Annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85

```
cw_extract_db_seqs cazy/cazyme.db genbank uniprot --classes GH,CE,CBM --kingdoms␣
↪bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

8.12.8 Providing a list of accessions

Instead of extracting protein sequences for all CAZymes matching a defined set of criteria, `cw_extract_db_seqs` can extract protein sequences a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_extract_db_seqs` a list of GenBank accessions to identify the specific set of CAZymes to extract protein sequences for.

The flag `--uniprot_accessions` can be used to provide `cw_extract_db_seqs` a list of UniProt accessions to identify the specific set of CAZymes to extract protein sequences for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_extract_db_seqs` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

The `sources` of the proteins operates independently of the `--genbank_accessions` and `--uniprot_accessions` flags. Therefore, the `--uniprot_accessions` flag can be used to identify a set of CAZymes of interest by their UniProt accession, and their protein sequence `source` can be defined as 'genbank', which will retrieve the GenBank protein sequence for the specified CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_extract_db_seqs` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_extract_db_seqs` will ignore the `--classes` flag and only extract protein squences for the proteins listed in the file provided via the `--genbank_accessions`.

8.13 Retrieving structure files from PDB

`cazy_webscraper` can be used to retrieve protein structure files for PDB accessions in a local CAZyme database from RSCB PDB database. The downloading of the structure files is handled by the `BioPython` module `Bio.PDB`.

For specific information of the `Bio.PDB` module please see the BioPython documentation.

Warning: If many PDB structure files are going to retrieved PDB (for example more than 100), it is expected practise to perform the opretional outside peak times.

Note: PDB structure files are retrieved for the PDB accessions that are *in* a local CAZyme database created using `cazy_webscraper`. A freshly built CAZyme database only contains NCBI protein accessions, taxonomic kingdoms, source organisms, and CAZy family annotations. Therefore, the `cw_get_uniprot_data` command must be used to retrieve PDB accessions from the UniProt database **prior** to using the `cw_get_pdb_structures` command.

8.13.1 Quick Start

To download the protein structure file for all PDB accessions in a local CAZyme database, use the following command structure:

```
cw_get_pdb_structures <path to local CAZyme db> <desired file formats>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

8.13.2 Structure file formats

`cw_get_pdb_structures` can retrieve protein structure files in a series of file formats. The options of file format are (as specified in the BioPython documentation):

- mmCif (default, PDBx/mmCif file),
- pdb (format PDB),
- xml (PDBML/XML format),
- mmtf (highly compressed),
- bundle (PDB formatted archive for large structure}

Any combination of file formats can be provided to `cw_get_pdb_structures` to download every file type for each PDB accession in the local CAZyme database. To list multiple file formats, separate each file format with a single space (' '). For example, to download the mmCif and xml files for every PDB accession in a local CAZyme database (located at `cazy/cazyme_db.db`), use the following command:

```
cw_get_pdb_structures cazy/cazyme_db.db mmCif xml
```

Warning: The file formats are case sensitive. For example, make sure to use 'mmCif' not 'mmcif'.

8.13.3 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`pdb` - **REQUIRED** List of file formats to retrieve from PDB for each PDB accession.

`--batch_size` - Size of an individual batch query of PDB accessions submitted to PDB. Default is 150.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec_filter` - List of EC numbers to limit the retrieval of structure files to proteins with at least one of the given EC numbers *in the local CAZyme database*.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete`, `-n` - When called, content in the existing output will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--outdir`, `-o` - Define output directory to write out structure files. Default, write structure files to current working directory.

`--overwrite` - Overwrite existing structure files with the same PDB accession as files being downloaded. Default false, do not overwrite existing files.

`--retries`, `-r` - Define the number of times to retry making a connection to PDB if the connection should fail. Default: 10.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--timeout`, `-t` - Connection timout limit (seconds). Default: 45.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to retrieve protein data for. A unique accession per line.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.13.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of protein structure files from PDB for proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which structure files are retrieved, others (e.g. `--species`) are used to filter and fine-tune the protein dataset for which structure files are retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy and UniProt, as retrieving data from PDB. Examples of using these flags can be found in the `cazy_webscraper` and `cw_get_uniprot_data` tutorial in this documentation.

Note: To retrieve protein structures for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.13.5 Structure file retrieval from PDB

The command for using `cazy_webscraper` for retrieval of PDB structure files from PDB is `cw_get_pdb_structures`.

8.14 Tutorials on configuring `cazy_webscraper` to retrieve data from PDB

`cazy_webscraper` can be configured to retrieve protein structures files user specified sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

`BioPython` is used to perform the retrieval of protein structure files from PDB. The retrieved structure files are written to the disk, *not* into the local CAZyme database.

> Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., … others. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422–1423.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of protein structured from PDB. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve PDB data call it using `cw_get_pdb_structures` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/pdb/get_pdb_structures.py`.

Note: PDB structure files are retrieved for the PDB accessions that are *in* a local CAZyme database created using `cazy_webscraper`. A freshly built CAZyme database only contains NCBI protein accessions, taxonomic kingdoms, source organisms, and CAZy family annotations. Therefore, the `cw_get_uniprot_data` command must be used to retrieve PDB accessions from the UniProt database **prior** to using the `cw_get_pdb_structures` command.

From this point on, we will be discussed the `cw_get_pdb_structures`, which is the entry point for retrieving data from PDB. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

8.14.1 Configuration via the command line

`cw_get_pdb_structures` has two required argument: * The path to the local CAZyme database created using `cazy_webscraper` * The structure file formats to retrieve the data from PDB in

The accepted structure file formats are: * mmCif (default, PDBx/mmCif file), * pdb (format PDB), * xml (PDBML/XML format), * mmtf (highly compressed), * bundle (PDB formatted archive for large structure}

Any combination of file formats can be provided to `cw_get_pdb_structures` to download every file type for each PDB accession in the local CAZyme database. To list multiple file formats, separate each file format with a single space (' '). For example, to download the mmCif and xml files for every PDB accession in a local CAZyme database (located at `cazy/cazyme_db.db`), use the following command:

```
cw_get_pdb_structures cazy/cazyme_db.db mmCif xml
```

Warning: The file formats are case sensitive. For example, make sure to use 'mmCif' not 'mmcif'.

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: Retrieve protein data for **all** CAZymes in the local CAZyme db

8.14.2 Options configurable at the command line

CAZymes of interest can be defined via providing:

- A set of GenBank accessions
- A set of UniProt accessions
- CAZy classes
- CAZy families
- Taxonomic kingdoms
- Genera
- Species
- Strains
- EC numbers

Here you can find a full list of the command-line flags and options.

8.14.3 Retrieving protein structures for specific CAZy classes and families

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving protein structure files from PDB.

For instance, if instead of retrieving protein data for all CAZymes in your local CAZyme database, you want to retrieve protein data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_pdb_structures cazy/cazyme.db mmCif --classes GH,CE
```

OR

```
cw_get_pdb_structures cazy/cazyme.db mmCif --classes Glycoside Hydrolases,Carbohydrate␣
↪Esterases
```

Retrieving protein data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein data for all proteins in PL1, PL2 and PL3 in the local CAZyme database, in mmCif and PDB format, use the following command:

```
cw_get_pdb_structures cazy/cazyme.db mmCif pdb --families PL1,PL2,PL3
```

Warning: `cw_get_pdb_structures` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_pdb_structures cazy/cazyme.db mmCif pdb --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cw_get_pdb_structures cazy/cazyme.db mmCif pdb --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

8.14.4 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have protein data retrieved from PDB.

For example, if you want to retrieve protein data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species, in pdb and xml formats, then use the command

```
cw_get_pdb_structures cazy/cazyme.db pdb xml --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve protein data for all CAZymes in a local CAZyme database that are derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_pdb_structures` command. The command we would use would be:

```
cw_get_pdb_structures cazy/cazyme.db pdb xml --kingdoms viruses --genera Aspergillus --
↪species Layia carnosa,Layia chrysanthemoides --strains Trichoderma reesei QM6a,
↪Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_pdb_structures` will retrieval CAZymes from *all* strains of the species.

8.14.5 Applying EC number filter

The retrieval of protein data from PDB can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations from UniProt and adding them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_pdb_structures cazy/cazyme.db pdb --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cw_get_pdb_structures` will retrieve the PDB structure files for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** PDB structure files will only be retrieved for CAZymes annotated for all provided EC numbers.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cw_get_pdb_structures` to retrieve data for protein A. This is because `cw_get_pdb_structures` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cw_get_pdb_structures` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.14.6 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species`, and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom protein structure files will be retrieved from PDB.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To retrieve PDB structure file for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species, and retrieve the files in mmCif and xml format we use the command:

```
cw_get_pdb_structures cazy/cazyme.db mmCif xml --classes GH,CE --families CE1,CE5,CE8 --
↪kingdoms bacteria
```

Example 2: To retrieve PDB structure files for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species in bundle format we use the command:

```
cw_get_pdb_structures cazy/cazyme.db bundle --classes GH --genera Aspegillus,Trichoderma
```

Example 3: To retrieve PDB structure files for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, and retrieve the files in pdb and xml format we use the command:

```
cw_get_pdb_structures cazy/cazyme.db xml pdb --classes GH,CE,CBM --kingdoms bacteria --
↪ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

Note: The order the structure file formats are provided does **not** matter.

8.14.7 Providing a list of accessions

Instead of retrieving protein structure files for all CAZymes matching a defined set of criteria, `cw_get_pdb_structures` can retrieve protein structure files for a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_get_pdb_structures` a list of GenBank accessions to identify the specific set of CAZymes to retrieve protein structure files for.

The flag `--uniprot_accessions` can be used to provide `cw_get_pdb_structures` a list of UniProt accessions to identify the specific set of CAZymes to retrieve protein structure files for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_pdb_structures` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_get_pdb_structures` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_pdb_structures` will ignore the `--classes` flag and only retrieve PDB structure files for the proteins listed in the file provided via the `--genbank_accessions`.

8.15 Retrieving NCBI Taxonomic Classifications

`cazy_webscraper` can be used to retrieve the latest taxonomic classification from the NCBI Taxonomy database for a set of proteins of interest in a local CAZyme database.

Querying NCBI is handled by the BioPython module `Bio.Entrez`.

8.15.1 Quick Start

To download the NCBI taxonomic classifications for all proteins in a local CAZyme datbase, use the following command structure:

```
cw_get_ncbi_taxs <path to local CAZyme db> <user email address>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

Note: The user email address is a requirement of NCBI.Entrez.

8.15.2 Storing the taxonomic classifications

The NCBI taxonomy classifications retrieved from the NCBI Taxonomy database are stored in the `NcbiTaxs` table in the local CAZyme database.

Each unique organism strain retrieved from NCBI is stored as a unique record in the `NcbiTaxs` table, which lists for each record the: * Superkingdom (referred to as the kingdom) * Phylum * Class (called tax_class in the database due to keyword clash with Python) * Order (called tax_order in the database due to keyword clash with SLQ) * Family * Genus * Species * Strain

The child prteins for each taxonomy record in the `NcbiTaxs` table is identified by the including a `ncbi_tax_id` from the `NcbiTaxs` table in the respecitve `Genbanks` table records.

8.15.3 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`email` - **REQUIRED** User email address. Required by NCBI.

`--batch_size` - Size of an individual batch query of NCBI sequence version accessions to NCBI. Default is 150.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec_filter` - List of EC numbers to limit the retrieval of structure files to proteins with at least one of the given EC numbers *in the local CAZyme database*.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--retries`, `-r` - Define the number of times to retry making a connection to CAZy if the connection should fail. Default: 10.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to retrieve protein data for. A unique accession per line.

`--update_gbk` - Update the existing NCBI taxonomy data in records in the `Genbanks` table already with NCBI taxonomy data. By default, NCBI tax data is only added to records in the `Genbanks` table if NCBI taxonomy data is not already presented in the record.

`--update_taxs` - Update existing NCBI taxonomy data in the `NcbiTaxs` table. By default onlt add new NCBI taxonomy data, do not update (and thus overwrite) existing data.

`--use_lineage_cache` - Use cached lineage data previously compliled by `cazy_webscraper` - skips retrieving NCBI Tax and Protein IDs and lineage data from NCBI

`--use_protein_ids` - Path to plain text file containing a tab delimited list of (1) NCBI Protein ID and (2) NCBI sequence version accession. Used cached NCBI Protein IDs.

`--use_tax_ids` - Path to plain text file listing a unique NCBI Taxonomy ID per line. Get lineages for cached NCBI Tax IDs.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.15.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of the NCBI taxonomic classifications for proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which taxonomic data will be retrieved. Others filters (e.g. `--species`) are used to filter and fine-tune the protein dataset for which data is retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy, UniProt, and PDB. Examples of using these flags can be found in the `cazy_webscraper` and `cw_get_uniprot_data` tutorial in this documentation.

Note: To retrieve taxonomic information for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.15.5 Retrieval of NCBI taxonomic classifications

The command for using `cazy_webscraper` for retrieving taxonomic classifications from the NCBI Taxonomy database is `cw_get_ncbi_taxs`.

8.16 Tutorials on configuring `cazy_webscraper` to retrieve NCBI taxonomic classifications

`cazy_webscraper` can be configured to retrieve the latest taxonomic classifications from the NCBI Taxonomy database for user specified sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

`BioPython` is used to perform the retrieval of taxonomic data from the NCBI Taxonomy database. The retrieved taxonomic classifications are stored in the local CAZyme database `NcbiTaxs` table. The child prteins for each taxonomy record in the `NcbiTaxs` table is identified by the including a `ncbi_tax_id` from the `NcbiTaxs` table in the respecitve `Genbanks` table records.

> Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., … others. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422–1423.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of taxonomic classifications from NCBI. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_get_ncbi_taxs` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/genbank/taxonomy/get_ncbi_taxs.py`.

From this point on, we will be discussed the `cw_get_ncbi_taxs`, which is the entry point for retrieving data from NCBI Taxonomy. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

8.16.1 Configuration via the command line

`cw_get_ncbi_taxs` has two required argument: * The path to the local CAZyme database created using `cazy_webscraper` * The user email address (required by NCBI)

```
cw_get_ncbi_taxs cazy/cazyme_db.db dummyEmail@domain.com
```

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: Retrieve the latest taxonomic classification from NCBI for all proteins in the local CAZyme database which do not currently have NCBI taxonomy data listed in local database, and the taxonomic information (i.e. the higher lineage classifications in the local database) are not updated.

8.16.2 Options configurable at the command line

CAZymes of interest can be defined via providing:

- A set of GenBank accessions
- A set of UniProt accessions
- CAZy classes
- CAZy families
- Taxonomic kingdoms
- Genera
- Species
- Strains
- EC numbers (if previously retrieved from UniProt)

Here you can find a full list of the command-line flags and options.

8.16.3 Retrieving taxonomy classifications for specific CAZy classes and families

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving taxonomy data from NCBI.

For instance, if instead of retrieving protein data for all CAZymes in your local CAZyme database, you want to retrieve protein data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes GH,CE
```

OR

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes Glycoside Hydrolases,
↪Carbohydrate Esterases
```

Retrieving protein data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein data for all proteins in PL1, PL2 and PL3 in the local CAZyme database, use the following command:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --families PL1,PL2,PL3
```

Warning: `cw_get_ncbi_taxs` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --families PL1,PL2,PL3 --classes␣
↪GH,CE
```

AND

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes GH,CE --families PL1,PL2,
↪PL3
```

are accepted.

8.16.4 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have data retrieved from NCBI taxonomy.

For example, if you want to retrieve data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species, then use the command

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve taxonomic data for all CAZymes in a local CAZyme database that are derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_ncbi_taxs` command. The command we would use would be:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --kingdoms viruses --genera␣
↪Aspergillus --species Layia carnosa,Layia chrysanthemoides --strains Trichoderma␣
↪reesei QM6a,Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_ncbi_taxs` will retrieve taxonomic data from *all* strains of the species.

8.16.5 Applying EC number filter

The retrieval of taxonomic data from NCBI can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations from UniProt and adding them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cw_get_ncbi_taxs` will retrieve the NCBI taxonomic classification for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** taxonomic data will only be retrieved for CAZymes annotated for all provided EC numbers.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cw_get_ncbi_taxs` to retrieve data for protein A. This is because `cw_get_ncbi_taxs` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cw_get_ncbi_taxs` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.16.6 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom taxonomic data will be retrieved from NCBI.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To taxonomic data for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species, we use the command:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes GH,CE --families CE1,CE5,
↪CE8 --kingdoms bacteria
```

Example 2: To taxonomic data for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species, we use the command:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes GH --genera Aspegillus,
↪Trichoderma
```

Example 3: To taxonomic classifications for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_ncbi_taxs cazy/cazyme.db dummyEmail@domain.com --classes GH,CE,CBM --kingdoms␣
↪bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

Note: The order the structure file formats are provided does **not** matter.

8.16.7 Providing a list of accessions

Instead of retrieving taxonomic data for all CAZymes matching a defined set of criteria, `cw_get_ncbi_taxs` can retrieve taxonomic data for a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_get_ncbi_taxs` a list of GenBank accessions to identify the specific set of CAZymes to retrieve taxonomic data for.

The flag `--uniprot_accessions` can be used to provide `cw_get_ncbi_taxs` a list of UniProt accessions to identify the specific set of CAZymes to retrieve taxonomic data for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_ncbi_taxs` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_get_ncbi_taxs` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_ncbi_taxs` will ignore the `--classes` flag and only taxonomic classifications for the proteins listed in the file provided via the `--genbank_accessions`.

8.17 Retrieving genomic assembly data from NCBI Assembly

`cazy_webscraper` can be used to retrieve the latest genomic assembly data from the NCBI Assembly database for a set of proteins of interest in a local CAZyme database.

Querying NCBI is handled by the BioPython module `Bio.Entrez`.

8.17.1 Quick Start

To download the genomic assembly data for all proteins in a local CAZyme datbase, use the following command structure:

```
cw_get_genomics <path to local CAZyme db> <user email address>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

Note: The user email address is a requirement of NCBI.Entrez.

8.17.2 Storing the taxonomic classifications

The NCBI genomic assembly data retrieved from the NCBI Assembly database are stored in the `Genomes` table in the local CAZyme database.

Specifically, `cazy_webscraper` retrieves the sequence version accession from the `Genbanks` table in the local CAZyme database that meet the user's specified criteria. The sequence accessions are then split into GenBank version accessions and RefSeq version accessions.

For GenBank version accessions the following data is retrieved from NCBI Assembly: * Assembly name * GenBank version accession * GenBank assembly ID * RefSeq version accession (if available) * RefSeq assembly ID (if available)

For RefSeq version accessions, the following data is retrieved from NCBI Assembly: * Assembly name * Refseq version accession * RefSeq assembly ID

Protein sequence accessions in the `Genbanks` table are linked to their source genomic assembly via the `Genbanks_Genomes` table.

To link protein sequence accessions with their source genomic assembly, `cazy_webscraper` queries NCBI Assembly to retrieve the genomic assembly data for all genomes linked to any of the protein sequence accessions retrieved from the local CAZyme database. `cazy_webscraper` then downloads the feature table for each genomic assembly, and parses the feature table to identify the CAZymes associated with the genomic assembly. The downloaded feature table is left in its compressed format and cached.

8.17.3 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`email` - **REQUIRED** User email address. Required by NCBI.

`--batch_size` - Size of an individual batch query of NCBI sequence version accessions to NCBI. Default is 150.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec_filter` - List of EC numbers to limit the retrieval of structure files to proteins with at least one of the given EC numbers *in the local CAZyme database*.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--retries`, `-r` - Define the number of times to retry making a connection to NCBI if the connection should fail. Default: 10.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to retrieve protein data for. A unique accession per line.

`--update` - Update the existing genomic assembly data in records in the `Genomes` table. By default, only NCBI assembly data not already in the local CAZyme database is added and existing data is not updated. Note updating data in the local CAZyme database `Genomes` table will overwrite existing data.

`--timeout` - Connection timeout limit (s) when downloading feature tables from NCBI Assembly. Default 45s)

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.17.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of the NCBI taxonomic classifications for proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which taxonomic data will be retrieved. Others filters (e.g. `--species`) are used to filter and fine-tune the protein dataset for which data is retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy, UniProt, and PDB. Examples of using these flags can be found in the `cazy_webscraper` and `cw_get_uniprot_data` tutorial in this documentation.

Note: To retrieve taxonomic information for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.17.5 Retrieval of NCBI genomic assembly data

The command for using `cazy_webscraper` for retrieving GenBank and RefSeq genomic assembly data from the NCBI Assembly database is `cw_get_genomics`.

8.18 Tutorials on configuring `cazy_webscraper` to retrieve NCBI genomic assembly data

`cazy_webscraper` can be configured to retrieve the latest genomic assembly data from the NCBI Assembly database for user specified sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of genomic assembly data from CAZy, UniProt, GenBank and PDB.

`BioPython` is used to perform the retrieval of genomic assembly data from the NCBI Assembly database. The retrieved genomic assembly data are stored in the local CAZyme database `Genomes` table. NCBI protein sequence version accessions in the local CAZyme database are linked to their source genomic assembly via the `Genbanks_Genomes` table.

> Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., … others. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11), 1422–1423.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of genomic assembly data from NCBI. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_get_genomics` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/genbank/genomes/get_genome_accs.py`.

From this point on, we will be discussed the `cw_get_genomics`, which is the entry point for retrieving data from NCBI Assembly. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

Specifically, the following data is retrieved from GenBank protein sequence version accessions: * Assembly name * GenBank version accession * GenBank assembly ID * RefSeq version accession (if available) * RefSeq assembly ID (if available)

For RefSeq version accessions, the following data is retrieved from NCBI Assembly: * Assembly name * Refseq version accession * RefSeq assembly ID

8.18.1 Configuration via the command line

`cw_get_genomics` has two required argument: * The path to the local CAZyme database created using `cazy_webscraper` * The user email address (required by NCBI)

```
cw_get_genomics cazy/cazyme_db.db dummyEmail@domain.com
```

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: Retrieve the latest genomic assembly data from NCBI for all proteins in the local CAZyme database which do not currently have NCBI genomic assembly data listed in local database, and the taxonomic information (i.e. the higher lineage classifications in the local database) are not updated.

8.18.2 Options configurable at the command line

CAZymes of interest can be defined via providing:

- A set of GenBank accessions
- A set of UniProt accessions
- CAZy classes
- CAZy families
- Taxonomic kingdoms
- Genera
- Species
- Strains

Here you can find a full list of the command-line flags and options.

8.18.3 Retrieving genomic assembly data for specific CAZy classes and families

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving genomic assembly data from NCBI.

For instance, if instead of retrieving genomic assembly data for all CAZymes in your local CAZyme database, you want to retrieve genomic assembly data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve genomic assembly data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve genomic assembly data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes GH,CE
```

OR

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes Glycoside Hydrolases,
↪Carbohydrate Esterases
```

Retrieving genomic assembly data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve genomic assembly data for all proteins in PL1, PL2 and PL3 in the local CAZyme database, use the following command:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --families PL1,PL2,PL3
```

Warning: `cw_get_genomics` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve genomic assembly data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --families PL1,PL2,PL3 --classes GH,
↪CE
```

AND

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes GH,CE --families PL1,PL2,
↪PL3
```

are accepted.

8.18.4 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve genomic assembly data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have data retrieved from NCBI taxonomy.

For example, if you want to retrieve data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species, then use the command

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve genomic assembly data for all CAZymes in a local CAZyme database that are derived from all

viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_genomics` command. The command we would use would be:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --kingdoms viruses --genera␣
↪Aspergillus --species Layia carnosa,Layia chrysanthemoides --strains Trichoderma␣
↪reesei QM6a,Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_genomics` will retrieve genomic assembly data from *all* strains of the species.

8.18.5 Applying EC number filter

The retrieval of genomic assembly data from NCBI can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations from UniProt and adding them to the local CAZyme database, you may wish to retrieve genomic assembly data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cw_get_genomics` will retrieve the NCBI genomic assembly data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** genomic assembly data will only be retrieved for CAZymes annotated for all provided EC numbers.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cw_get_genomics` to retrieve data for protein A. This is because `cw_get_genomics` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cw_get_genomics` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.18.6 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom genomic assembly data will be retrieved from NCBI.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To genomic assembly data for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species, we use the command:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes GH,CE --families CE1,CE5,
↪CE8 --kingdoms bacteria
```

Example 2: To genomic assembly data for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species, we use the command:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes GH --genera Aspegillus,
↪Trichoderma
```

Example 3: To genomic assembly data for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_genomics cazy/cazyme.db dummyEmail@domain.com --classes GH,CE,CBM --kingdoms␣
↪bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

Note: The order the structure file formats are provided does **not** matter.

8.18.7 Providing a list of accessions

Instead of retrieving genomic assembly data for all CAZymes matching a defined set of criteria, `cw_get_genomics` can retrieve genomic assembly data for a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_get_genomics` a list of GenBank accessions to identify the specific set of CAZymes to retrieve genomic assembly data for.

The flag `--uniprot_accessions` can be used to provide `cw_get_genomics` a list of UniProt accessions to identify the specific set of CAZymes to retrieve genomic assembly data for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_genomics` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_get_genomics` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_genomics` will ignore the `--classes` flag and only genomic assembly data for the proteins listed in the file provided via the `--genbank_accessions`.

8.19 Retrieving GTDB Taxonomic Classifications

`cazy_webscraper` can be used to retrieve the latest taxonomic classification from the Genome Taxonomy Database (GTDB) taxonomy database for a set of proteins of interest in a local CAZyme database.

Note: As in the GTDB database, GTDB taxonomic classifications are retrieved and associated with genomes stored in the local CAZyme database. To retrieve GTDB taxonomic classifications the genomic data for the proteins of interest **must** be listed in the local CAZyme database.

8.19.1 Quick Start

To download the GTDB taxonomic classifications for all proteins in a local CAZyme datbase, use the following command structure:

```
cw_get_gtdb_taxs <path to local CAZyme db> <kingdoms>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

Note: GTDB only provides data for bacteria and archaeal genomes.

8.19.2 Storing the taxonomic classifications

The GTDB taxonomy classifications retrieved from the GTDB Taxonomy database are stored in the `GtdbTaxs` table in the local CAZyme database.

Each unique organism strain retrieved from GTDB is stored as a unique record in the `GtdbTaxs` table, which lists for each record the: * Superkingdom (referred to as the kingdom) * Phylum * Class (called tax_class in the database due to keyword clash with Python) * Order (called tax_order in the database due to keyword clash with SLQ) * Family * Genus * Species * Strain * Release - the GTDB release from which the lineage was retrieved

The child prteins for each taxonomy record in the `GtdbTaxs` table is identified by the including a `gtdb_tax_id` from the `GtdbTaxs` table in the respecitve `Genomes` table records.

8.19.3 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`taxs` - **REQUIRED** Kingdoms to get lineages from. Accepts 'archaea' and/or 'bacteria'. Separate with a single space. Order does not matter. Determines which datafiles are retrieved from GTDB.

`--archaea_file` - Path to GTDB archaea data file. Default: None, download latest dataset from GTDB.

`--bacteria_file` - Path to GTDB bacteria data file. Default: None, download latest dataset from GTDB.

Note: The filenames of provided GTDB data files must match the filename format used by GTDB, to allow `cazy_webscraper` to retrieve the release number of the dataset.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve UniProt data for.

`--ec_filter` - List of EC numbers to limit the retrieval of structure files to proteins with at least one of the given EC numbers *in the local CAZyme database*.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--retries`, `-r` - Define the number of times to retry making a connection to GTDB if the connection should fail. Default: 10.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to retrieve protein data for. A unique accession per line.

`--update_genome_lineage_gbk` - Update Genome GTDB lineage. Default: only add lineages to Genomes without a lineage.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.19.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of the GTDB taxonomic classifications for proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which taxonomic data will be retrieved. Others filters (e.g. `--species`) are used to filter and fine-tune the protein dataset for which data is retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy, UniProt, and PDB. Examples of using these flags can be found in the `cazy_webscraper` and `cw_get_uniprot_data` tutorial in this documentation.

Note: To retrieve taxonomic information for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.19.5 Retrieval of GTDB taxonomic classifications

The command for using `cazy_webscraper` for retrieving taxonomic classifications from the GTDB Taxonomy database is `cw_get_gtdb_taxs`.

8.20 Tutorials on configuring `cazy_webscraper` to retrieve GTDB taxonomic classifications

`cazy_webscraper` can be configured to retrieve the latest taxonomic classifications from the Genome Taxonomy Database (GTDB) for user specified sets of CAZymes in a local CAZyme database. Many of the same configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

The retrieved taxonomic classifications are stored in the local CAZyme database `GtdbTaxs` table. The child prteins for each taxonomy record in the `GtdbTaxs` table is identified by the including a `ncbi_tax_id` from the `GtdbTaxs` table in the respecitve `Genomes` table records.

Note: As in the GTDB database, GTDB taxonomic classifications are retrieved and associated with genomes stored in the local CAZyme database. To retrieve GTDB taxonomic classifications the genomic data for the proteins of interest **must** be listed in the local CAZyme database.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of taxonomic classifications from GTDB. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_get_gtdb_taxs` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/expand/genbank/taxonomy/get_ncbi_taxs.py`.

From this point on, we will be discussed the `cw_get_gtdb_taxs`, which is the entry point for retrieving data from GTDB Taxonomy. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

8.20.1 Configuration via the command line

`cw_get_gtdb_taxs` has two required arguments: * The path to the local CAZyme database created using `cazy_webscraper` * Source kingdoms. Accepts 'archaea' and/or 'bacteria'

```
cw_get_gtdb_taxs cazy/cazyme_db.db archaea bacteria
```

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: Retrieve the latest taxonomic classification from GTDB for all proteins in the local CAZyme database which do not currently have GTDB taxonomy data listed in local database, and the taxonomic information (i.e. the higher lineage classifications in the local database) are not updated.

8.20.2 Selecting source kingdoms

GTDB catalogues the taxonomic lineages of archaea and bacterial species. Defining the source taxonomic kingdom(s) (the second positional argument for `cw_get_gtdb_taxs`) determines which datafiles are retrieved from GTDB, and thus which taxonomic lineages are added to the local CAZyme database. This is separate to the `--kingdoms` filter which is used to define CAZymes of interest by the taxonomic classification retrieved from CAZy.

To add only archaeal lineages retrieved from GTDB use only `archaea`:

```
cw_get_gtdb_taxs cazy/cazyme_db.db archaea
```

To add only bacterial lineages to the local CAZyme database, using only `bacteria`:

```
cw_get_gtdb_taxs cazy/cazyme_db.db bacteria
```

To add both archaeal and bacterial lineages from GTDB to the local CAZyme database using both `archaea` and `bacteria` (in any order), separated with a singel space:

```
cw_get_gtdb_taxs cazy/cazyme_db.db bacteria archaea
```

8.20.3 Options configurable at the command line

CAZymes of interest can be defined via providing:

- A set of GenBank accessions
- A set of UniProt accessions
- CAZy classes
- CAZy families

- Taxonomic kingdoms
- Genera
- Species
- Strains
- EC numbers (if previously retrieved from UniProt)

Here you can find a full list of the command-line flags and options.

8.20.4 Retrieving taxonomy classifications for specific CAZy classes and families

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving taxonomy data from GTDB.

For instance, if instead of retrieving protein data for all CAZymes in your local CAZyme database, you want to retrieve protein data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases then use the command:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes GH,CE
```

OR

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes Glycoside Hydrolases,
↪Carbohydrate Esterases
```

Retrieving protein data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein data for all proteins in PL1, PL2 and PL3 in the local CAZyme database, use the following command:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --families PL1,PL2,PL3
```

Warning: `cw_get_gtdb_taxs` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

8.20.5 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have data retrieved from GTDB taxonomy.

For example, if you want to retrieve data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species, then use the command

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve taxonomic data for all CAZymes in a local CAZyme database that are derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_get_gtdb_taxs` command. The command we would use would be:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --kingdoms viruses --genera Aspergillus␣
↪--species Layia carnosa,Layia chrysanthemoides --strains Trichoderma reesei QM6a,
↪Trichoderma reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_get_gtdb_taxs` will retrieve taxonomic data from *all* strains of the species.

8.20.6 Applying EC number filter

The retrieval of taxonomic data from GTDB can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations from UniProt and adding them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cw_get_gtdb_taxs` will retrieve the GTDB taxonomic classification for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** taxonomic data will only be retrieved for CAZymes annotated for all provided EC numbers.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cw_get_gtdb_taxs` to retrieve data for protein A. This is because `cw_get_gtdb_taxs` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cw_get_gtdb_taxs` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.20.7 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom taxonomic data will be retrieved from GTDB.

Below we run through 3 example commands of combining these flags, and the resulting behaviour.

Example 1: To add taxonomic data for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species, we use the command:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes GH,CE --families CE1,CE5,CE8 -
↪-kingdoms bacteria
```

Example 2: To add taxonomic data for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species, we use the command:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes GH --genera Aspegillus,
↪Trichoderma
```

Example 3: To add taxonomic classifications for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_gtdb_taxs cazy/cazyme.db archaea bacteria --classes GH,CE,CBM --kingdoms bacteria␣
↪--ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

Example 4: To add bacterial taxonomic classifications for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_get_gtdb_taxs cazy/cazyme.db bacteria --classes GH,CE,CBM --kingdoms bacteria --ec_
↪filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

8.20.8 Providing a list of accessions

Instead of retrieving taxonomic data for all CAZymes matching a defined set of criteria, `cw_get_gtdb_taxs` can retrieve taxonomic data for a set of CAZymes defined by their GenBank and/or UniProt accession.

The flag `--genbank_accessions` can be used to provide `cw_get_gtdb_taxs` a list of GenBank accessions to identify the specific set of CAZymes to retrieve taxonomic data for.

The flag `--uniprot_accessions` can be used to provide `cw_get_gtdb_taxs` a list of UniProt accessions to identify the specific set of CAZymes to retrieve taxonomic data for.

In both instances (for `--genbank_accessions` and `--uniprot_accessions`) the list of respective accessions are provided via a plain text file, with a unique protein accession of each line. The path to this file is then passed to `cw_get_gtdb_taxs` via the respective `--genbank_accessions` and `--uniprot_accessions` flag.

`--genbank_accessions` and `--uniprot_accessions` can be used at the same time to define all CAZymes of interest.

Warning: `--genbank_accessions` and `--uniprot_accessions` take president over the filter flags.

When either `--genbank_accessions` or `--uniprot_accessions` is used, `cw_get_gtdb_taxs` will **not** retrieve any CAZymes from the local database matching a set of criteria.

Therefore, if `--genbank_accessions` and `--classes` are used, `cw_get_gtdb_taxs` will ignore the `--classes` flag and only taxonomic classifications for the proteins listed in the file provided via the `--genbank_accessions`.

8.20.9 Providing GTDB datafiles

By default `--cazy_webscraper` retrieves the latest GTDB datafiles from the GTDB website. However, you can provide your own GTDB datafiles.

Specifically, these are the avaialble from GTDB release page.

The filenames must use the same filename format as GTDB to enable the correct extraction of the GTDB release number from the filename.

To provide a previously downloaded archaea datafile use the `--archaea_file` flag followed by the path point to the target data file.

Similarly, to provide a previously downloaded bacteria datafile use the `--bacteria_file` flag followed by the path point to the target data file.

Note: `cazy_webscraper` excepts GTDB datafiles in the compressed (`.gz`) tab-separated file format (`.tsv`).

8.20.10 Updating genomic classifications

By default `cw_get_gtdb_taxs` only adds links to GTDB lineages to genomes in the local CAZyme database that are not already linked to a GTDB lineage. To update which GTDB lineage genomes in the local CAZyme database are linked to add the `--update_genome_lineage` flag.

```
cw_get_gtdb_taxs cazy/cazyme.db bacteria \
   --classes GH,CE,CBM \
   --kingdoms bacteria \
   --update_genome_lineage
```

8.21 Interrogating the data using the API

The data stored in the local CAZyme database can be interrogated using SQL. `cazy_webscraper` also includes an API, which can be used to interrogate the data in the local CAZyme database and write out the retrieved data in `JSON` and/or `CSV` format.

By default `cazy_webscraper` only includes the GenBank accessions of proteins matching the provided criteria, but the inclusion of additional data (such as protein squences, UniProt accessions, EC numbers, etc) is fully customisable.

8.21.1 Quick Start

To retrieve the GenBank accession of all CAZymes stored in the local CAZyme database, use the following command structure:

```
cw_query_database <path to local CAZyme db> <desired file formats>
```

Note: The `cw` prefix on command is an abbreviation of `cazy_webscraper`.

8.21.2 Accepted file formats

`cw_query_database` can write the output to a csv or json file.

These are provided as the second arguments to `cw_query_database`. To write out both a csv and json file use both `csv` and `json` after the path to the local CAZyme database, separted with a single space.

```
cw_query_database <path to local CAZyme db> csv json
```

Note: The order `csv` and `json` are written does not matter.

Warning: Both `csv` and `json` are case sensitive.

8.21.3 Command line options

`database` - **REQUIRED** Path to a local CAZyme database to add UniProt data to.

`file_types` - **REQUIRED** List of file formats to export the data in. Currently supported: `csv` and `json`.

`--cache_dir` - Path to cache dir to be used instead of default cache dir path.

`--cazy_synonyms` - Path to a JSON file containing accepted CAZy class synonsyms if the default are not sufficient.

`--config`, `-c` - Path to a configuration YAML file. Default: None.

`--classes` - list of classes to retrieve data from.

`--ec_filter` - List of EC numbers to limit the retrieval of protein data with at least one of the given EC numbers *in the local CAZyme database*.

`--force`, `-f` - Force writing to existing output dir. Default: False, do not write to existing target output dir. Not applied when `--output_dir` is not called.

`--families` - List of CAZy (sub)families to retrieve UniProt protein data for.

`--genbank_accessions` - Path to text file containing a list of GenBank accessions to retrieve protein data for. A unique accession per line.

`--genera` - List of genera to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given genera.

`--include` - List additional data to include in the output. Multiple fields can be named, separating each with a single space (' '). The accepted fields are: * 'class' - Include the CAZy class annotations * 'family' - Include the CAZy family annotations * 'subfamily' - Include the subfamily class annotations * 'kingdom' - Include the taxonomic kingdom of the source organism * 'genus' - Include the genus of the source organism * 'organism' - Include the scientific name of the source organism * 'uniprot_acc' - Include the UniProt accession * 'uniprot_name' - Include the protein name retrieved from UniProt * 'ec' - Include the EC number annotations * 'pdb' - Include the PDB accessions * 'genbank_seq' - Include the GenBank protein sequence * 'uniprot_seq' - Include the Uniprot protein sequence

`--kingdoms` - List of taxonomy kingdoms to retrieve UniProt data for.

`--log`, `-l` - Target path to write out a log file. If not called, no log file is written. Default: None (no log file is written out).

`--nodelete`, `-n` - When called, content in the existing output will **not** be deleted. Default: False (existing content is deleted).

`--nodelete_cache` - When called, content in the existing cache dir will **not** be deleted. Default: False (existing content is deleted).

`--output_dir`, `-o` - Define output directory to write out structure files. Default, write structure files to current working directory.

`--overwrite` - Overwrite existing structure files with the same PDB accession as files being downloaded. Default false, do not overwrite existing files.

`--prefix`, `-p` - Had prefix to all output files.

`--retries`, `-r` - Define the number of times to retry making a connection to CAZy if the connection should fail. Default: 10.

`--sql_echo` - Set SQLite engine echo parameter to True, causing SQLite to print log messages. Default: False.

`--species` - List of species (organsim scientific names) to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given species.

`--strains` - List of species strains to restrict the retrieval of protein to data from UniProt to proteins belonging to one of the given strains.

`--timeout`, `-t` - Connection timout limit (seconds). Default: 45.

`--uniprot_accessions` - Path to text file containing a list of UniProt accessions to retrieve protein data for. A unique accession per line.

`--verbose`, `-v` - Enable verbose logging. This does **not** set the SQLite engine `echo` parameter to True. Default: False.

8.21.4 Basic Usage

The command-line options listed above can be used in combination to customise the retrieval of data for proteins of interest. Some options (e.g. `--families` and `--classes`) define the broad group of proteins for which structure files are retrieved, others (e.g. `--species`) are used to filter and fine-tune the protein dataset for which structure files are retrieved.

The `--classes`, `--families`, `--kingdoms`, `--genera`, `--species`, and `--strains` filteres are applied in the exactly same for retrieving data from CAZy, UniProt, GenBank, and PDB as for the extraction of data via the API. Examples of using these flags can be found in the `cazy_webscraper` and `cw_query_database` tutorial in this documentation.

Note: To retrieve fata for members of specific CAZy subfamilies, list the subfamilies after the `--families` flag.

8.21.5 Retrieving data from a local CAZyme database

The command for using `cazy_webscraper` for interrogating the local CAZyme database and extract data is `cw_query_databsae`.

8.22 Tutorial on interrogating the data using the API

`cazy_webscraper` includes an API that can be used to interrogate the data in the local CAZyme database and write out the retrieved data in `JSON` and/or `CSV` format.

By default `cazy_webscraper` only includes the GenBank accessions of proteins matching the provided criteria, but the inclusion of additional data (such as protein squences, UniProt accessions, EC numbers, etc) is fully customisable.

For interrogating the data and retrieving proteins matching the provided criteria, many of the same configuration options apply to the retrieval of protein data from CAZy, UniProt, GenBank and PDB.

`cazy_webscraper` can be configured via the **command line** and/or via a **YAML configuration file**.

This page runs through examples of how to combine the various 'filters' that can be applied, to fully customised the retrieval of protein data from the local CAZyme database. These tutorials are designed for those with less experience using command-line tools.

Note: If you installed `cazy_webscraper` using `bioconda` or `pip` to invoke `cazy_webscraper` to retrieve UniProt data call it using `cw_query_database` - this is the method used in this tutorial. If you installed `cazy_webscraper` from source then you will need to invoke `cazy_webscraper` from the root of the repo using the command `python3 cazy_webscraper/api/cw_query_database.py`.

From this point on, we will be discusseing the `cw_query_database` command, which is used by `cazy_webscraper` for retrieving protein data from the local CAZyme database. We also presume you are comfortable configuring `cazy_webscraper` for the scraping of data from CAZy.

8.22.1 Configuration via the command line

`cw_query_database` requires two arguments: * The path to the local CAZyme database created using `cazy_webscraper` * The file formats to write out the output

Therefore, `cw_query_database` can be enabled using a simple command structure:

```
cazy_webscraper <path to the local CAZyme db> <file formats>
```

For example, if our database was stored in `cazy/cazyme.db` and we want to write the output to a csv file, we would used:

```
cazy_webscraper cazy/cazyme.db csv
```

Note: Make sure `cw_query_database` is pointed directly at the database file.

When no optional arguments are provided, the default behaviour is invoked. The default behaviour is to: retrieve only the GenBank accessions of **all** CAZymes in the local CAZyme CAZyme db

8.22.2 Accepted file formats

`cw_query_database` can write the output to a csv or json file.

These are provided as the second arguments to `cw_query_database`. To write out both a csv and json file use both `csv` and `json` after the path to the local CAZyme database, separted with a single space.

```
cw_query_database <path to local CAZyme db> csv json
```

Note: The order `csv` and `json` are written does not matter.

Warning: Both `csv` and `json` are case sensitive.

8.22.3 Options configurable at the command line

The following behaviours of the `cw_query_database` can be configured at the command-line in the terminal:

- Limit the retrieval of protein data to CAZymes in the local databaes from specific CAZy classes, CAZy families, kingdoms, genuera, species, strains and/or EC numbers
- **Including any combination of the following in the output, along side the GenBank accessions:**
  - CAZy class
  - CAZy family
  - CAZy subfamily
  - Kingdom
  - Genus
  - Scientific name of the source organsim
  - Protein sequence retrieved from GenBank
  - UniProt accession
  - Protein name retrieved from UniProt
  - EC numbers
  - PDB accessions
  - Protein sequence retrieved from UniProt
- Write the output in JSON and/or CSV format
- Choose an output directory
- Force overwriting existing files
- Enable verbose logging during the operation of the webscraper

Here you can find a full list of the command-line flags and options.

8.22.4 Choosing an output directory

By default, `cw_query_database` writes all output files to the current working directory.

To specify an alternative output directory, using the `--output_dir/-o` flag, followed by the path to the target output directory. `cw_query_database` will build all necessary parent and child output directories.

For example, to write the output to the directory `my_cazy_data` use the following command:

If the output directory already exists, `cw_query_database` will raise an error warning the output directory already exists and close. This is to prevent accidently writing data to the wrong location.

To force `cw_query_database` to write the data to an existing output directory, add the `--force/-f` flag.

By default `cw_query_database` will delete all content already present in the existing output directory. To retain the data in the existing output directory, add the `--nodelete/-n` flag.

Note: The `--force` and `--nodelete` flags are only applied when the `--output_dir` flag is used. `cw_query_database` will **not** delete content in the current working directory when writing to the current working directory when the `--output_dir` flag is **not** used.

8.22.5 Overwrite existing files

`cw_query_database` automatically compiles the names of the output files.

The file names of all output files are composed of the name of the local CAZyme database, followed by the names of the data retrieved from the local CAZyme database. For example, retrieving the following data from the local CAZyme database called `cazy_database.db`: * CAZy family annotation * CAZy subfamily annotations * EC numbers * PDB accessions Will produce the following file name: `cazy_database_gbkAcc_fams_subfams_ec_pdb`.

Note: `_gbkAcc` is always included in the file name because GenBank accessions are always retrieved and written to the output by `cw_query_database`.

Both the *json* and *csv* files are given the same name, the files only differ in their file extension.

An optional prefix can be applied to all output file names using the `-p/--prefix` flag, followed by the desired prefix. For example, using the same example as above, the prefix 'engineering_candidates_` can be applied to every output file by adding the following to command:

This will produce output files with the file name `engineering_candidates_cazy_database_fams_subfams_ec_pdb`.

If files matching the file names compiled by `cw_query_database` already existing at the target output location, `cw_query_database` will raise a warning that output files already existing and terminate. This is to prevent accidently overwriting data files.

To overwrite existing datafiles add the `--overwrite` flag to the command. For example, the following command will retrieve all GenBank accessions stored in the local CAZyme database located at `cazy/cazyme.db` and write out the GenBank accessions to a file called `all_gbk_acc_cazyme_gbkAcc.csv` to `my_cazy_data`, and will not delete content in the existing output directory and will overwrite the existing output file `my_cazy_data/all_gbk_acc_cazyme_gbkAcc.csv`.

8.22.6 Retrieving protein data for CAZy classes and families to scrape

The `--classes` and `--families` flags from scraping data from CAZy are applied in the extact same way for retrieving protein data from the local CAZyme databases.

For instance, if instead of retrieving protein data for all CAZymes in your local CAZyme database, you want to retrieve protein data for CAZymes in specific CAZy classes then add the `--classes` flag followed by the classes you want to retrieve protein data for.

Tip: To list multiple classes, separate the classes with a single comma.

For example, if you want to retrieve protein data for all CAZymes from Glycoside Hydrolase and Carbohydrate Esterases, and write the data to a csv file, then use the command:

```
cw_query_database cazy/cazyme.db csv --classes GH,CE
```

OR

```
cw_query_database cazy/cazyme.db csv --classes Glycoside Hydrolases,Carbohydrate␣
↪Esterases
```

Retrieving protein data for proteins from specific specific CAZy families is achieved using the `--families` flag. For example, to retrieve protein data for all proteins in PL1, PL2 and PL3 in the local CAZyme database, and write the data to csv and json files, use the following command:

```
cw_query_database cazy/cazyme.db json csv --families PL1,PL2,PL3
```

Warning: `cw_query_database` only accpets families written in the proper CAZy family syntax. GH1 is accepted. gh1 and GlycosideHydrolases1 are not accepted.

As with scraping data from CAZy, the `--classes` and `--families` flags can be combined. To retrieve protein data for all CAZymes in PL1, PL2, PL3 and *all* of GH and CE both, and write the data to a json file:

```
cw_query_database cazy/cazyme.db json --families PL1,PL2,PL3 --classes GH,CE
```

AND

```
cw_query_database cazy/cazyme.db json --classes GH,CE --families PL1,PL2,PL3
```

are accepted.

8.22.7 Applying taxonomic

The `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used to refine the dataset of proteins to retrieve protein data by taxonomy. These flags are applied in the exact same way as they are used for the scraping of data from CAZy. Only proteins in the local CAZyme database and matching at least on of the provided taxonomy criteria will have protein data retrieved from GenBank and added to the local CAZyme datbase.

For example, if you want to retrieve protein data for all CAZymes in a local CAZyme database from bacterial and eukaryotic species then use the command

```
cw_query_database cazy/cazyme.db csv --kingdoms bacteria,eukaryota
```

Warning: The kingdoms must be spelt the same way CAZy spells them, for example use 'eukaryot**a**' instead of 'eukaryot**e**'.

Note: The kingdoms are **not** case sensitive, therefore, both `bacteria` *and* `Bacteria` are accepted.

Note: You can list the kingdoms in *any* order. Thus, both `bacteria,eukaryota` *and* `eukaryota,bacteria` are accepted.

You can combine any combination of the optional flags, including combining the taxonomic filters. For example, you may wish to retrieve protein data for all CAZymes in a local CAZyme database that are derived from all viral species, Aspergillus species, Layia carnosa, Layia chrysanthemoides, Trichoderma reesei QM6a and Trichoderma reesei QM9414. To do this we would combine the respective flags for a single `cw_query_database` command. The command we would use would be:

```
cw_query_database cazy/cazyme.db csv --kingdoms viruses --genera Aspergillus --species␣
↪Layia carnosa,Layia chrysanthemoides --strains Trichoderma reesei QM6a,Trichoderma␣
↪reesei QM9414
```

Note: The order that the flags are used and the order taxa are listed does **not** matter, and separate multiple taxa names with a single comma with **no** spaces.

Warning: Use the standard scientific name formating. Captialise the first letter of *genus* and write a lower case letter for the first letter of the species.

Aspergillus niger is **correct**

asepergillus niger is **incorrect**

ASPERGILLUS NIGER is **incorrect**

Warning: When you specify a species `cw_query_database` will retrieval CAZymes from *all* strains of the species.

8.22.8 Applying EC number filter

The retrieval of protein data from the local CAZyme database can also be limited to proteins in a local CAZyme database that are annotated with specific EC numbers.

Having previously retrieved EC number annotations and added them to the local CAZyme database, you may wish to retrieve protein data for CAZymes annotated with specific EC numbers. To do this add the `--ec_filter` flag to the command, follwed by a list of EC numbers.

```
cw_query_database cazy/cazyme.db csv --ec_filter "EC1.2.3.4,EC2.3.4.5"
```

Note: Provide complete EC numbers. Both dashes ('-') and asterixes ('*') are accepted for missing digits in EC numbers.

EC1.2.3.- and EC1.2.3.* are accepted. EC1.2.3. and EC 1.2.3 are **not** accepted.

Note: The 'EC' prefix is not necessary. EC1.2.3.4 and 1.2.3.4 are accepted.

Warning: If using dashes to represent missing digits in EC numbers, it is recommended to bookend the entire EC number list in single or double quotation marks. Some terminals may misinterpret EC1.2.-.- as trying to invoke the options '.'

Note: `cazy_webscraper` will retrieve the specified UniProt data for all proteins in the local CAZyme database that are annotated with **at least one** of the given EC numbers. Therefore, if multiple EC numbers are given this **does not mean** only CAZymes will all provided EC numbers will have data retrieved from UniProt for them.

`--ec_filter` is based upon EC number annotations stored within the local CAZyme database. For example, if protein A is annotated with the EC1.2.3.4, but this annotation is not stored in the local CAZyme database, using `--ec_filter EC1.2.3.4` will **not** cause `cazy_webscraper` to retrieve data for protein A. This is because `cazy_webscraper` does not know protein A is annotated with EC1.2.3.4, because this annotation is not within its database.

Warning: If `--ec_filter` is used along side `--ec`, `cazy_webscraper` will retrieve **all** EC number annotations from UniProt for all proteins in the local CAZyme database that are associated with at least one of the EC numbers provided via `--ec_filter` within the CAZyme database.

8.22.9 Combining all filters

The `--classes`, `--families`, `--ec_filter`, `--kingdoms`, `--genera`, `--species` and `--strains` flags can be used in any combination to define a specific subset of proteins in the local CAZyme database for whom protein data from GenBank will be retrieved.

Below we run through 3 example commands of combining these flags, writing the output to a csv file, and the resulting behaviour.

Example 1: To retrieve protein data for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species we use the command:

```
cw_query_database cazy/cazyme.db csv --classes GH,CE --families CE1,CE5,CE8 --kingdoms␣
↪bacteria
```

Example 2: To protein data for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species we use the command:

```
cw_query_database cazy/cazyme.db csv -classes GH --genera Aspegillus,Trichoderma
```

Example 3: To retrieve protein data for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, we use the command:

```
cw_query_database cazy/cazyme.db csv --ec --sequences --classes GH,CE,CBM --kingdoms␣
↪bacteria --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85"
```

8.22.10 Customising the output

By defauly `cw_query_database` only includes the GenBank accessions of the CAZymes matching the provided criteria in the final output. Any combination of the following can also be included in the output: * CAZy class * CAZy family * CAZy subfamily * Kingdom * Genus * Scientific name of the source organsim * Protein sequence retrieved from GenBank * UniProt accession * Protein name retrieved from UniProt * EC numbers * PDB accessions * Protein sequence retrieved from UniProt

To include additional data in the output use the `--include` flag followed by any combination (and any order) of the following options: `--include` - List additional data to include in the output. Multiple fields can be named, separating each with a single space (' '). The accepted fields are: * 'class' - Include the CAZy class annotations * 'family' - Include the CAZy family annotations * 'subfamily' - Include the subfamily class annotations * 'kingdom' - Include the taxonomic kingdom of the source organism * 'genus' - Include the genus of the source organism * 'organism' - Include the scientific name of the source organism * 'uniprot_acc' - Include the UniProt accession * 'uniprot_name' - Include the protein name retrieved from UniProt * 'ec' - Include the EC number annotations * 'pdb' - Include the PDB accessions * 'genbank_seq' - Include the GenBank protein sequence * 'uniprot_seq' - Include the Uniprot protein sequence

---

Note: The quotation marks around the terms do not need to be included.

Note: No matter what additional data is included in the output, the data will be presented in the same order as presented above. For example, 'Kingdom' will always come before all fields listed below it. Changing the order the fields are listed in the command will not change the order data is presented in the output. For example, using `--include kingdom genus organism` and `--include organism genus kingdom` will both result in the respective columns being placed in the order of: 'Kingdom', 'Genus', 'Organism'

To list multiple fields to include in the final output, separate each field with a singel space (' ').

Example 1: To retrieve protein data for all CAZymes in GH, GT, CE1, CE5 and CE8, and which are derived from baceterial species, and include the CAZy family annotations and scientific names of the source organisms we use the command:

```
cw_query_database cazy/cazyme.db csv --classes GH,CE --families CE1,CE5,CE8 --kingdoms␣
↪bacteria --include family organism
```

Example 2: To protein data for all CAZymes in GH and which are derived from *Aspegillus* and *Trichoderma* species, and include the CAZy class, EC number and PDB accessions in the output we use the command:

```
cw_query_database cazy/cazyme.db csv --include class ec pdb --classes GH --genera␣
↪Aspegillus,Trichoderma
```

Example 3: To retrieve protein data for all CAZymes in GH,CE and CBM which are derived from baceterial species and are annotated with at least one of EC3.2.1.23, EC3.2.1.37 and EC3.2.1.85, and include the EC number annotations, CAZy family and CAZy subfamily annotations we use the command:

```
cw_query_database cazy/cazyme.db csv \
   --ec --sequences \
   --classes GH,CE,CBM \
   --kingdoms bacteria \
   --ec_filter "3.2.1.23,3.2.1.37,3.2.1.85" \
   --include family subfamily ec
```

8.23 Caching and Using the Cache

To facilitate the reproducibility of scrapping CAZy, `cazy_webscraper` logs each scrape within the compiled CAZyme datbase and writes out cache files.

This pages walks through the data logged in the CAZyme database and the different cache files written by `cazy_webscraper`, as well as how to define a different cache directory, instead of the default.

8.23.1 The Logs table

The database built by `cazy_webscraper` contains a table called 'Logs'. This table logs every scrape of CAZy, UniProt and GenBank which added data to the database.

The table contains the following columns and data:

- **log_id:** Autoincrement ID number
- **date:** Date scrape was initated (in ISO format)
- **time:** Time scrape was initated (in ISO format)
- **database:** Name of the external database from which data was retrieved (i.e. 'CAZy', 'UniProt' or 'GenBank')
- **retrieved_annotations:** List of annotation types retrieved (e.g. 'EC number, PDB accession, Sequence')
- **classes:** CAZy classes for which data was retrieved
- **families:** CAZy families for which data was retrievedclasses
- **kingdoms:** taxonomy Kingdoms filteres applied
- **genera_filter:** taxonomy Kingdoms filteres applied
- **species_filter:** taxonomy Kingdoms filteres applied
- **strains_filter:** taxonomy Kingdoms filteres applied
- **ec_filter:** EC fliters applied to retrieve data for CAZymes annotated with the specified EC numbers (only applies to retrieval of data from UniProt, GenBank and PDB)
- **cmd_line:** Reproduction of the command line arguments passed to `cazy_webscraper`.

The 'Logs' table allows any one who uses the database to see how the dataset was compiled.

8.23.2 Cache files when retrieving data from CAZy

`cazy_webscraper` writes out 2 cache files. These are:

- `cazy_db_<date-time>.zip` which is the txt file downloaded from CAZy containing all CAZy Data
- `<db_name>_<date-time>_connection_failures.log` which contains a list of proteins for which data was unsuccessfully parsed, and CAZy families for which the family member population could not be retrieved (if the 'validation' option is used).

8.23.3 Cache files when retrieving data from UniProt

The dataframes retrieved with each query to UniProt are cached.

In addition two JSON files are created: * `uniprot_accessions_YYYY-MM-DD_HH-MM-SS.json` * `uniprot_data_YYYY-MM-DD_HH-MM-SS.json`

UniProt accessions JSON file

The first file (`uniprot_accessions`) contains the UniProt accessions/IDs for each GenBank accession retrieved from the local CAZyme database that matches the provided criteria. These UniProt IDs are used to query UniProt and retrieve protein data. UniProt cannot be batch queried by GenBank accessions to retrieve protein data using `bioservices`.

The JSON file is keyed by the UniProt accession and is valued by a Python dictionary like structure, containing the GenBank accession the corresponding ID of its record in the local CAZyme database. For example:

This file can be used to skip the retrieval of UniProt IDs from UniProt (which is the first step performed by `cw_get_uniprot_data`). To do this use the `--skip_uniprot_accessions` flag followed by the path to the corresponding `uniprot_accessions_YYYY-MM-DD_HH-MM-SS.json` file.

UniProt data JSON file

The second json file (`uniprot_data`) contains all data retrieved from UniProt for all proteins in the local CAZyme database that match the specified criteria. The data retrieved from UniProt was parsed into a Python dictionary which is then dumped into the JSON file.

This file is used for mannually checking the parsing method employed by `cw_get_uniprot_data` is working, as well as skipping the retrieval of the same dataset from UniProt (for example, if you wanted to recreate a specific CAZyme proteome dataset).

To use the data cached in the `uniprot_data` file, using the `--use_uniprot_cache` flag, followed by the path pointing to the corresponding file. Using this flag, skips the retrieval of protein data from UniProt, and only adds data from the cache file into the local CAZyme database.

8.23.4 Cache files when retrieving protein sequences from GenBank

When retrieving protein sequences from GenBank, all cache files will be contained in the `genbank_seq_retrieval.tar` compressed file. The cache files that will be generated are:

three cache file are produced:

- `cw_genbanks_seqs_<cache_time>.fasta` contains the protein sequences downloaded from GenBank
- `failed_retrieve_ids` contains the GenBank accessions containing the IDs of **all** proteins for whom no protein sequence was retrieved

- `failed_entrez_connection_accessions` contains the GenBank accessions of proteins whom no protein sequence was retrieved owing to failure to connect to NCBI
- `invalid_ids` contains GenBank protein version accessions that were retrieved from CAZy, but are no longer listed in NCBI

Note: `cazy_webscraper` includes the option to use cached protein sequences, to (i) skip the retrieval of data from GenBank and (ii) facilitate the reproduction of local CAZyme databases.

8.23.5 Cache files when retrieving taxonomic information from NCBI

When retrieving taxonomic classifications the following cache files are generated: * `failed_protein_accs.txt` - list the version accession of all proteins for which not taxonomy data could be retrieved from NCBI (possibly the protein record has been removed) * `lineage_data.json` - lists the lineage data and the associated protein sequences accessions * `ncbi_lineages.json` - lists the lineage data retrieved from NCBI Taxonomy * `protein_ncbi_ids.out` - the NCBI Protein IDs retrieved from NCBI when querying by protein version accession * `tax_ids.out` - the NCBI Taxonomy IDs retrieved from NCBI when querying by NCBI Protein IDs

When retrieving genomic assembly data a file listing all protein sequence accessions for which no data was retrieved is cached (`failed_protein__accessions.txt`). Additionally, the downloaded genomic assemlby feature tables are retained in their compress format and cached.

8.23.6 Cache files when retrieving data from PDB

One cache file is created when using `cazy_webscraper` to retrieval protein structure files from PDB: `pdb_retrieval_YYYY-MM-DD_HH-MM-SS.txt`, which lists the PDB accessions of all files that were successfully downloaded from PDB using `cazy_webscraper` and `BioPython`.

8.23.7 Cache files when retrieving genomic information from NCBI

All genomic assembly feature tables are cached in the cache directory. In addition, two other cache files are generated to list proteins and genomes for whom data could not be retrieved from NCBI:

- `no_assembly_urls.txt` contains the NCBI Assembly Name of genomic assemlbies for whom a feature table could not be downloaded from the NCBI FTTP server - typically because a feature table is not available
- `failed_protein_accessions.txt` contains the NCBI protein version accessions for whom genomic assembly information

8.23.8 Cache files when retrieving taxonomic information from GTDB

The GTDB database dumbs are downloaded and written to the cache directory:

- `archaea_data-{release_number}.gz`
- `bacteria_data-{release_number}.gz`

8.23.9 Cache directory

By default `cazy_webscraper` creates the cache directory in the same directory that the datbase is created, and with the name `.cazy_webscraper`.

To use a different cache directory instead add the `--cache_dir` flag, followed by the path to the cache directory.

Note: The cache directory does not need to already exist, `cazy_webscraper` will build the cache directory and all it's parent directories.

If the target cache directory already exists, by default `cazy_webscraper` will delete the content already present in the already existing cache directory. To not delete the exsiting directory content add the `nodelete_cache` flag.

8.24 Integrate a local CAZyme database into into downstream analyses

To facilitate integrating the dataset compiled using `cazy_webscraper`, the data downloaded from external databases is compiled into a database using the wildly used local SQLite3 database format.

To facilitate integratting the local CAZyme database into third-party `Python` applications, use the `get_db_connection` function from `cazy_webscraper`, which will return an open connection to the CAZyme database from `sqlalchemy`.

`get_db_connection` takes 2 required args and one optional arg: **Required:** - Path to the local CAZyme database (provided as a `pathlib.Path` object) - Bool to set the `sqlalchemy.create_engine` param `sql_echo`. When set to `True`, the SQL log will be printed to the terminal **Optional:** - Bool to reflect if a new database. Default is `False`, i.e. connecting to an existing local CAZyme database

Import the function into the `Python` script using:

```
from cazy_webscraper.sql.sql_orm import get_db_connection
```

8.25 Contributing

8.25.1 Reporting bugs and errors

If you find a bug, or an error in the code or documentation, please report this by raising an issue at the GitHub issues page for `cazy_webscraper`

8.25.2 Contributing code or documentation

We gratefully accept code contributions, if you would like to fix a bug, improve documentation, or extend `cazy_webscraper` and its subcommands. To make everyone's lives easier in this process, we ask that you please follow the procedure below:

1. Fork the `cazy_webscraper` repository under your account at GitHub.
2. Clone your fork to your development machine.
3. Create a new branch in your forked repository with an informative name, e.g. `git checkout -b fix_issue_107`.

4. Make the changes.
5. Run the unit tests in the repository.
6. If the tests pass, push the changes to your fork, and submit a pull request against the original repository.
7. Indicate one of the `cazy_webscraper` developers as an assignee in your pull request.

8.25.3 Suggestions for improvement

If you would like to make a suggestion for how we could improve `cazy_webscraper`, we welcome contributions at the GitHub issues page.

8.26 Citations: cite cazy_webscraper and dependencies

8.26.1 Cite `cazy_webscraper`

If you use `cazy_webscraper`, please cite the following publication:

> Hobbs, Emma E. M.; Pritchard, Leighton; Chapman, Sean; Gloster, Tracey M. (2021): cazy_webscraper Microbiology Society Annual Conference 2021 poster. FigShare. Poster. [https://doi.org/10.6084/m9.figshare.14370860.v7{]}(https://doi.org/10.6084/m9.figshare.14370860.v7)

8.26.2 Citing dependencies

cazy_webscraper depends on a number of tools. To recognise the contributions that the authors and developers have made, please also cite the following:

When making an SQLite database:

> Hipp, R. D. (2020) SQLite, available: https://www.sqlite.org/index.html.

Retrieving taxonomic, genomic or sequence data from NCBI:

> Cock, P.J.A., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., et al (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics, Bioinformatics, 25(11), 1422-1423.
>
> Wheeler,D.L., Benson,D.A., Bryant,S., Canese,K., Church,D.M., Edgar,R., Federhen,S., Helmberg,W., Kenton,D., Khovayko,O. et al (2005) Database resources of the National Centre for Biotechnology Information: Update, Nucleic Acid Research, 33, D39-D45

Retrieving data from UniProt:

> Cokelaer, T., Pultz, D., Harder, L. M., Serra-Musach, J., Saez-Rodriguez, J. (2013) BioServices: a common Python package to access biological Web Services programmatically, Bioinformatics, 19(24), 3241-3242.

Downloading protein structure files from RSCB PDB:

> Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., et al (2022) The Protein Data Bank, Nucleic Acids Research, 28(1), 235-242.
>
> Hamelryck, T., Manderick, B. (2003), PDB parser and structure class implemented in Python. Bioinformatics, 19 (17), 2308–2310

Retrieving and using taxonomic data from GTDB:

Parks, D.H., Chuvochina, M., Rinke, C., Mussig, A.J., Chaumeil, P., Hugenholtz, P. (2022) GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy, Nucleic Acids Research, 50(D1), D785-D794.

8.27 License

8.27.1 MIT License

8.28 Questions?

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

CHAPTER

NINE

CITING CAZY_WEBSCRAPER

If you use `cazy_webscraper` in your work *please* do cite our work (including the provided DOI), as well as the specific version of the tool you use. This is not only helpful for us as developers to get our work out into the world, but it is also **essential for the reproducibility and integrity of scientific research**. Citation:

> Hobbs, Emma E. M.; Pritchard, Leighton; Chapman, Sean; Gloster, Tracey M. (2021): cazy_webscraper Microbiology Society Annual Conference 2021 poster. FigShare. Poster. https://doi.org/10.6084/m9.figshare.14370860.v7

`cazy_webscraper` depends on a number of tools. To recognise the contributions that the authors and developers have made, please also cite the following:

When making an SQLite database:

Hipp, R. D. (2020) SQLite, available: https://www.sqlite.org/index.html.

Retrieving taxonomic, genomic or sequence data from NCBI:

Cock, P.J.A., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., et al (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics, Bioinformatics, 25(11), 1422-1423.

Wheeler,D.L., Benson,D.A., Bryant,S., Canese,K., Church,D.M., Edgar,R., Federhen,S., Helmberg,W., Kenton,D., Khovayko,O. et al (2005) Database resources of the National Centre for Biotechnology Information: Update, Nucleic Acid Research, 33, D39-D45

Retrieving data from UniProt:

Cokelaer, T., Pultz, D., Harder, L. M., Serra-Musach, J., Saez-Rodriguez, J. (2013) BioServices: a common Python package to access biological Web Services programmatically, Bioinformatics, 19(24), 3241-3242.

Downloading protein structure files from RSCB PDB:

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., et al (2022) The Protein Data Bank, Nucleic Acids Research, 28(1), 235-242.

Hamelryck, T., Manderick, B. (2003), PDB parser and structure class implemented in Python. Bioinformatics, 19 (17), 2308–2310

Retrieving and using taxonomic data from GTDB:

Parks, D.H., Chuvochina, M., Rinke, C., Mussig, A.J., Chaumeil, P., Hugenholtz, P. (2022) GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy, Nucleic Acids Research, 50(D1), D785-D794.

CHAPTER

TEN

DEVELOPMENT AND ISSUES

If there are additional features you wish to be added, you have problems with the scraper, or would like to contribute please raise an issue at the GitHub repository.

- Issues Page: https://github.com/HobnobMancer/cazy_webscraper/issues