ProteoClade

Release 0.0.1

Apr 01, 2020

Introduction

1 Requirements **3**
- 1.1 Software . . . 3
- 1.2 Hardware . . . 3

2 Installation **5**

3 Questions/Comments? **7**

4 Getting Started **9**
- 4.1 Set Up Your Analysis . . . 9
- 4.2 Import ProteoClade . . . 9
- 4.3 Assemble Required Files . . . 9
  - 4.3.1 1. ProteoClade Database (PCDB) . . . 9
  - 4.3.2 2. The ProteoClade Taxonomy Mapping File (PCTAXA) . . . 10

5 Targeted Database Searches **11**
- 5.1 File Format Requirements . . . 11
- 5.2 Annotation . . . 11
- 5.3 Gene Level Quantitation . . . 12

6 *de novo* Searches **13**
- 6.1 File Format Requirements . . . 13
- 6.2 Annotation . . . 14

7 ProteoClade 5 Minute Demo **15**
- 7.1 Prepare Data . . . 15
- 7.2 Targeted Database Example . . . 15
- 7.3 *de novo* Example . . . 16

8 `pcutilities` Module **17**
- 8.1 download_uniprot . . . 17
- 8.2 download_uniprot_batch . . . 18
- 8.3 download_cRAP . . . 18
- 8.4 download_taxonomy . . . 18
- 8.5 load_taxonomy . . . 19
- 8.6 ncbi_check . . . 19
- 8.7 db_stats . . . 19

8.8 db_fetch_params . . . 19

9 `pcdb` Module **21**
9.1 create_pcdb . . . 21
9.2 merge_fastas . . . 22

10 `pcannotate` Module **23**
10.1 annotate_peptides . . . 23
10.2 annotate_denovo . . . 24
10.3 filter_taxa . . . 24

11 `pcquant` Module **25**
11.1 roll_up . . . 25

12 `pcconstants` Module **27**
12.1 cleave_rules . . . 27
12.2 uniprot_options . . . 27
12.3 ncbi_ranks . . . 27
12.4 valid_quant_cols . . . 28
12.5 valid_sample_cols . . . 28
12.6 valid_pep_cols . . . 28
12.7 valid_score_cols . . . 28
12.8 basic_annotations . . . 28
12.9 proteoclade_cols . . . 28

Python Module Index **29**

Index **31**

ProteoClade is a Python library for **taxonomic-based annotation and quantification of bottom-up proteomics data**. It is designed to be user-friendly, and has been optimized for speed and storage requirements.

ProteoClade helps you analyze two general categories of experiments:

1. **Targeted Database Searches:** Experiments in which a limited number of species are defined ahead of time, such as those involving Patient-Derived Xenografts (PDXs) or host-pathogen interactions. Reference protein sequence databases are used for targeted searches (ex: using Mascot, MaxQuant).
2. **De Novo Searches:** Experiments in which the organisms are unspecified ahead of time or involve samples of high taxonomic complexity. Mass spectra are analyzed in the absence of a reference database (ex: using PEAKS, PepNovo).

ProteoClade scales from two organisms to every organism in UniProt. This documentation includes examples from each kind of proteomic workflow to familiarize the user with ProteoClade's features.

Check out and cite the publication at PLOS Computational Biology.

Mooradian AD, van der Post S, Naegle KM, Held JM (2020) ProteoClade: A taxonomic toolkit for multi-species and metaproteomic analysis. PLOS Computational Biology 16(3): e1007741.

CHAPTER 1

Requirements

1.1 Software

ProteoClade's only dependency is Python 3.6+. It runs on Windows, MacOS, and Linux. Note that MacOS users will need to run the "Install Certificates.command" file in their Python 3 directory to enable access to the web.

1.2 Hardware

- CPU: 1 core (minimum); 4 cores (recommended)
- RAM: 8 gigabytes (minimum); 16 gigabytes (recommended)
- Hard drive: 1 gigabyte free (minimum); Solid State Drive (recommended)

CHAPTER 2

Installation

ProteoClade can be installed using pip:

```
pip install proteoclade
```

Or if you have multiple Python versions:

```
python3 -m pip install proteoclade
```

Or visit our github to download the package directly.

CHAPTER 3

Questions/Comments?

E-mail the Held lab:

Arshag: mooradian@wustl.edu

Jason: jheld@wustl.edu

CHAPTER 4

Getting Started

4.1 Set Up Your Analysis

Create a new folder to hold experimental files and results, and navigate to it in the command line of your operating system. This will be your **working directory**. ProteoClade will use this directory by default to look for sequence, taxonomy, and experimental files.

For experimental files, refer to the file format requirements for targeted searches or *de novo* searches.

4.2 Import ProteoClade

Common ProteoClade functions are most easily made available by importing everything from the package. Open a Python 3.6+ shell from the command line and then import:

```
>>> from proteoclade import *
```

4.3 Assemble Required Files

ProteoClade needs to generate two files:

4.3.1 1. ProteoClade Database (PCDB)

In order to efficiently map experimental peptides to taxa and genes, a SQLite database (.pcdb) is generated using FASTA-formatted files. ProteoClade facilitates this by retrieving user-specified taxa from UniProt.

Downloading FASTA files from UniProt can be accomplished using the function download_uniprot:

```
download_uniprot(*targets, download_folder = 'fastas')
```

Where targets are tuples containing (taxonomyID, database_type). The download folder will create a 'fastas' folder in your working directory if one does not already exist.

Example:

```
>>> download_uniprot((9606,'sr'),(10090,'sr'))
# Downloads the human and mouse Swiss-Prot reference proteomes
```

Alternatively, download_uniprot_batch can be used to read a tab-delimited file to download multiple entries.

Optional step: once FASTA files have been downloaded, users may elect to include additional FASTA files by dragging and dropping them into the same directory as any that were downloaded by ProteoClade. User-supplied FASTA files are expected to have headers adhering loosely to the UniProt format. Only three fields are required:

```
>db|UniqueIdentifier|EntryName OX=TaxonID GN=GeneName
```

Where TaxonID is the NCBI-assigned taxonomy identifier, and GeneName is the gene symbol. If no GeneName is found, the UniProt UniqueIdentifier is substituted.

Optional step: all FASTA files can be merged into a single file for database search engines using the function merge_fastas. This function will warn on the first incorrect header supplied but will not terminate. Note that incorrectly formatted headers will be discarded when a PCDB is created.

Finally, the PCDB file can be created using the create_pcdb function. ProteoClade enables the user to specify several parameters which should match the experimental, analytical, and informatics conditions of the experiment. An *in silico* digest is performed with these parameters so that experimentally-detected peptides can be matched. Please refer to the function's documentation for all of the parameters.

Example:

```
>>> create_pcdb('human_mouse_swissprot')
    # Creates a pcdb file using trypsin and all default parameters
```

4.3.2 2. The ProteoClade Taxonomy Mapping File (PCTAXA)

Annotating and taxa requires taxonomy mapping, which ProteoClade will source from NCBI. This is accomplished with the function download_taxonomy. Note that you should store the PCDB and PCTAXA files together, as taxonomy ID assignment may change over time. For ease of identification, contaminants are mapped with a TaxonID of -1 and a taxon name of "CONTAMINANT" at every level of the hierarchy, and this should be reflected in any contaminant FASTAs included (the function to download the cRAPome has this feature already built-in).

Example:

```
>>> download_taxonomy()
    #Retrieves and builds the PCTAXA file in the working directory, named by the date␣
↪it was retrieved
```

CHAPTER 5

Targeted Database Searches

Database searches typically generate result files containing unique peptide sequences and quantitative information which can be annotated and quantified for taxon-specific proteomics using ProteoClade.

5.1 File Format Requirements

ProteoClade uses generic tab-delimited or comma-delimited text files for annotation. The table only requires a single column (RED).

Sequence	sample1	sample2	othercol
PEPTIDER	13759	15301	

5.2 Annotation

The sequence column can be named "Sequence", "Peptide", or "pep_seq", and peptides should be represented as sequences in all capital letters (any other characters will be stripped prior to matching). Any number of additional columns can be included. These files are annotated with the function annotate_peptides:

```
annotate_peptides(file, db, pctaxa, taxon_levels = None, worker_threads = None)
```

Example:

```
>>> annotate_peptides('myinput.csv', 'hu_mou_sp.pcdb', '190101.pctaxa', taxon_levels␣
↪= 'species')
```

With default parameters, a file will have the following output, with multiple annotations for a given peptide separated by the pipe symbol (new columns: BLUE).

Sequence	sample1	sample2	othercol	organisms	genes
PEPTIDER	13759	15301		9606|10090	IPO5|Ipo5

5.3 Gene Level Quantitation

In order to represent the proteomics data as gene products rather than peptides, annotated peptide files can be summed to the gene level using the roll_up function. This produces a gene and taxon-specific list of genes with their relevant quantitative information. Note that the default taxon for annotation is "species", so make sure this is added when the annotate functions are called. For this process, there are two ways of signifying sample columns, which will be used for quantitation:

1. "implicit" : ProteoClade will use anything that is not known to be a ProteoClade used or generated column. Use command proteoclade_cols to see columns that ProteoClade will safely ignore.
2. "explicit" : ProteoClade will only select columns that begin with the text "sample_".

As ProteoClade is looking for samples to quantify in this step, **please do not use "intensity", or other names found in the 'proteoclade_cols' global variable, to name your sample.** These columns will be avoided in the roll_up step.

The following example input format, using the "implicit" method, shows the columns that ProteoClade knows to ignore in RED with sample columns implicitly found in PURPLE:

peptide	F7	F7_count	organisms	genes	Species
DVTSSSR	0	13	9606	KRT14	Homo sapiens
ATVEDEK	0	2	10090|9606	Hspa8|HSPA8	Mus musculus|Homo sapiens
SGTEVGR	0	5	9606	CRNN	Homo sapiens

Example:

```
roll_up("annotated_myinput.csv")
    # Roll up to genes if peptide belongs to a single species
```

The output shows the unique taxon that was assigned, the gene, a sample's quantitative value, and its spectral counts if derived from a PSM/de novo file. Note that for the sake of gene uniqueness, genes are converted to upper case:

Taxon	Gene	F7	F7_count
Homo sapiens	CRNN	0	9
Homo sapiens	AZGP1	0	11
Mus musculus	SYNE2	0	1

CHAPTER 6

de novo Searches

de novo searches yield peptide sequences without a reference database. ProteoClade enables the ability to annotate *de novo* search results to understand which organisms are present in the sample and what proteins they are expressing.

6.1 File Format Requirements

Files generated from *de novo* search engines, like PEAKS, are typically in long format rather than wide format – i.e., there is a single column for all samples. Candidate PSMs (multiple suggestions for one MS/MS scan) can be used as long as the file is sorted first by sample:scan, and second by score (in descending order). The required input format for these files is a .csv or .txt with the following columns (required: RED, optional: GREY):

Peptide	Scan	Score	Area
VTPER	F7:16	90	1500
TVPER	F7:16	82	1500
SAELNK	F7:6	88	1916
SAELGGK	F7:6	73	1916

The table must contain:

- Exactly one peptide column (options: Sequence, Peptide, pep_seq)
- Exactly one sample column (options: Sample, Scan). **Formatting for this column must be: Sample:Scan**.
- At most one score column (options: Score, ALC (%))
- At most one quantitation column (options: Area, Intensity)

6.2 Annotation

These files can be annotated in the same way as peptide files for targeted searches, but instead using the function annotate_denovo. The parameters are the same with one change: a PSM matching method option is added.

In order to preserve memory, this function generates a "denovo_matched" file containing only organisms and genes from the PCDB file and then invokes the annotate_peptides function on that file to add additional taxa, if specified in the *taxon_levels* parameter.

A typical output will include the peptide sequences, sample quantitation, spectral counts for that peptide, organisms, genes, and higher taxa if specified:

peptide	F7	F7_count	organisms	genes	species
DVTSSSR	0	13	9606	KRT14	Homo sapiens
ATVEDEK	0	2	10090|9606	Hspa8|HSPA8	Mus musculus|Homo sapiens
SGTEVGR	0	5	9606	CRNN	Homo sapiens

CHAPTER 7

ProteoClade 5 Minute Demo

Want to see ProteoClade in action? This tutorial provides a quick run through of both targeted and *de novo* workflows to demonstrate the tool's features.

7.1 Prepare Data

1. Install ProteoClade
2. Download some example data. Extract these files to a folder. "targeted_humouse_example.txt" is a truncated and reformated MaxQuant search from a patient-derived xenograft data set, while "denovo_bacteria_example.csv" is a truncated *de novo* PEAKS search from an oral microbiome data set.
3. Navigate to the folder with the example data, open a Python 3 shell, and import ProteoClade:

```
>>> from proteoclade import *
```

4. Download and assemble taxonomy information from the NCBI:

```
>>> download_taxonomy()
```

7.2 Targeted Database Example

1. Download protein sequence information from UniProt:

```
>>> download_uniprot((9606,'sr'),(10090,'sr'), download_folder = 'pdxseq')
    #Downloads human and mouse proteomes by Taxon ID.
```

2. Create a PCDB for patient-derived xenografts:

```
>>> create_pcdb('humouse', 'pdxseq')
```

3. Annotate the targeted experiment. Make sure to replace the XXXXXX with the date/name of the PCTAXA file you generated in step 4 of "Prepare Data".:

```
>>> annotate_peptides('targeted_humouse_example.txt', 'humouse.pcdb', 'XXXXXX.
↪pctaxa', taxon_levels = ('species','phylum'))
```

4. Roll up peptide information to gene symbols:

```
>>> roll_up('annotated_targeted_humouse_example.txt')
```

Results: "rollup_annotated_targeted_humouse_example.txt" now contains genes derived from species-specific peptides and their summed ion intensities.

7.3 *de novo* Example

1. Download protein sequence information from UniProt:

```
>>> download_uniprot((1891914,'a'),(1283313,'a'), download_folder = 'denovoseq')
    #Downloads strep oralis and alloprevotella proteomes by Taxon ID.
```

2. Create a PCDB for patient-derived xenografts:

```
>>> create_pcdb('bacteria', 'denovoseq')
```

3. Annotate the *de novo* experiment. Make sure to replace the XXXXXX with the date/name of the PCTAXA file you generated in step 4 of "Prepare Data".:

```
>>> annotate_denovo('denovo_bacteria_example.csv', 'bacteria.pcdb', 'XXXXXX.pctaxa
↪', taxon_levels = ('species','phylum'))
```

4. Roll up peptide information to gene symbols:

```
>>> roll_up('annotated_denovo_matched_denovo_bacteria_example.csv')
```

Results: 'annotated_denovo_matched_denovo_bacteria_example.csv' contains species and phyla annotations for the *de novo* data set, while 'rollup_annotated_denovo_matched_denovo_bacteria_example.csv' contains peptides summed to gene symbols. Note that although this *de novo* data set does not contain quantitative information, spectral counts are provided in additional columns.

CHAPTER 8

pcutilities Module

- *download_uniprot*
- *download_uniprot_batch*
- *download_cRAP*
- *download_taxonomy*
- *load_taxonomy*
- *ncbi_check*
- *db_stats*
- *db_fetch_params*

8.1 download_uniprot

proteoclade.pcutilities.**download_uniprot**(**targets*, *download_folder='fastas'*)

Download FASTA protein sequences from UniProt

Parameters

- **targets** (*tuple*) – One or more tuples containing an integer and a string (TaxonID, DBtype) TaxonID must be a NCBI-valid taxon identifier OR 'all' for all taxa (up to ~60GB) DBType must be one of: "s", "sr", "r", "t", "a"

  (SwissProt, SwissProt Reference, Reference, TrEMBL, or All, respectively)

- **download_folder** (*string*) – Folder which will contain downloaded FASTA files. Default: 'fasta' subdirectory

Examples

```
>>> download_uniprot((9606, 's'),(10090, 's')) #downloads human and mouse␣
↪SwissProt entries
>>> download_uniprot(('all','a')) #downloads every entry in UniProt
```

Notes

Downloads UniProt-derived FASTA file(s) with specified parameters. Naming convention: taxonid_StartingEntryCount.fasta

8.2 download_uniprot_batch

proteoclade.pcutilities.**download_uniprot_batch**(*file*, *download_folder='fastas'*)

Download UniProt entries from a tab delimited txt file

Parameters

- **file** (*string*) – tab-delimited txt file with 2 columns: [TaxonID,UniProtMods]

  Use the uniprot_options constant for available mods
- **download_folder** (*string*) – subdirectory to store downloads (default 'fastas')

Notes

Easier method to download larger numbers of taxa.

8.3 download_cRAP

proteoclade.pcutilities.**download_cRAP**(*directory='fastas'*)

Downloads contaminant Repository for Affinity Purification, (cRAP) sequence database.

Parameters **directory** (*string*) – folder to download cRAP to (default 'fastas')

Notes

Supplies an edited fasta file from cRAP

8.4 download_taxonomy

proteoclade.pcutilities.**download_taxonomy**(*directory='taxonomy_downloads'*)

Download taxonomy mappings from NCBI.

Parameters **directory** (*string*) – Where to store temporary files downloaded from NCBI (default 'taxonomy_downloads')

Notes

Unzips taxonomy files from NCBI. Will call _taxonomy_mapper to produce PCTAXA file in working directory. PCTAXA file is a pickled dict of taxonomy mapping. Naming will be Y-M-D formatted so you can remember when it was retrieved.

8.5 load_taxonomy

proteoclade.pcutilities.**load_taxonomy**(*file*)

Loads PCTAXA file into memory.

Parameters **file** (*string*) – A .pctaxa file created using the download_taxonomy function.

Returns

taxonomy dictionary – This dictionary contains all NCBI taxonomy mappings for an organism ID.

dictionary[TaxID] = {'species': species, 'genus': genus, ... }

Return type dict

Example

```
>>> taxonomy = load_taxonomy('190101.pctaxa')
>>> taxonomy[9606].get('species')
Homo sapiens
```

8.6 ncbi_check

proteoclade.pcutilities.**ncbi_check**(*taxa*)

Validates a list of taxa by making sure they are NCBI-valid ranks.

Parameters **taxa** (*tuple or list*) – List of taxonomic ranks, i.e. ('order','family')

Returns List of only valid taxonomic ranks.

Return type list

8.7 db_stats

proteoclade.pcutilities.**db_stats**(*db*)

Prints database parameters from database for the user.

Parameters **db** (*string*) – .pcdb file created with create_pcdb; prints out stats.

8.8 db_fetch_params

proteoclade.pcutilities.**db_fetch_params**(*db*)

Used in pcannotate.py but also may be useful for the user. Will retrieve the digest parameters specified when the database was created.

Parameters **db** (*string*) – PCDB (SQLite db) to connect to

Returns **results** – tuple of parameters (min_length, max_length, missed_cleavages, digest_rule, date_created). If parameters are not found, None is returned.

Return type tuple or None

CHAPTER 9

pcdb Module

- *create_pcdb*
- *merge_fastas*

9.1 create_pcdb

proteoclade.pcdb.**create_pcdb**(*database_name*, *fasta_directory='fastas'*, *min_length=7*, *max_length=55*, *missed_cleavages=2*, *m_cleave=True*, *li_swap=True*, *rule='trypsin/p'*, *temp_directory=None*, *worker_count=None*, *reverse=False*)

Creates the PCDB file which stores in silico digested peptides, genes, and organism info.

Parameters

- **database_name** (*string*) – Name of pcdb file; should be descriptive of what it contains
- **fasta_directory** (*string*) – Directory of FASTAs to use as input (default 'fastas')
- **min_length** (*integer*) – Minimum peptide amino acid count to include in database (default 7)
- **max_length** (*integer*) – Maximum peptide amino acid count to include in database (default 55)
- **missed_cleavages** (*integer*) – Number of times a protease is allowed to miss a cut site. (default 2)
- **m_cleave** (*bool*) – Whether or not N-terminal methionines are cleaved from proteins (default True)
- **li_swap** (*bool*) – Whether peptides stored will have leucines converted to isoleucines (default True)
- **rule** (*string or tuple*) – Protease rule for cutting sites (default 'trypsin/p').

  if string: must be an enzyme option available in ProteoClade. See Appendix.

if tuple: must be tuple of strings, ("regex_sites","terminus") ex. (r"[RK]", "C"). Use tuple for custom enzyme rules.

- **temp_directory** (*None or string*) – Directory for temporary database operations if space is a concern (default None)

  if None: uses working directory

- **worker_count** (*None or integer*) – Number of worker processes to use. Only set to experiment with performance. (default None) if None: determines processes up to a maximum of 6 to use. More processes does not necessarily increase performance.
- **reverse** (*bool*) – Whether to reverse protein sequences prior to digestion and storage. Used for FDR mitigation.

Examples

```
>>> create_pcdb("human_mouse_ref.pcdb") #creates a trypsin PCDB file using␣
↪default settings
>>> create_pcdb("bacteria_swissprot.pcdb", rule = "asp-n") #creates an AspN PCDB␣
↪file
```

Notes

Creates a .pcdb SQLite file in the working directory.

9.2 merge_fastas

proteoclade.pcdb.**merge_fastas**(*merged_fasta_name*, *fasta_directory='fastas'*)

For merging fastas together in a directory. Used in preparation of a targeted database search, i.e. MaxQuant/Mascot.

Parameters

- **merged_fasta_name** (*string*) – Name of .fasta file that will result from merging other fastas
- **fasta_directory** (*string*) – Directory from which to read fastas (default 'fastas')

Notes

Creates a .fasta file containing all read fasta entries.

CHAPTER 10

pcannotate Module

- *annotate_peptides*
- *annotate_denovo*
- *filter_taxa*

10.1 annotate_peptides

proteoclade.pcannotate.**annotate_peptides**(*file*, *db*, *pctaxa*, *taxon_levels=None*, *worker_threads=None*)

Drives the taxonomic and gene annotation of peptide-containing files.

Parameters

- **file** (*string*) – csv or txt file containing wide-form, peptide entries
- **db** (*string*) – PCDB file containing digested peptides to match w/ experiment
- **pctaxa** (*string*) – PCTAXA file containing taxonomic mapping for species and above
- **taxon_levels** (*None, string, or tuple*) – Which taxa to annotate above the organism level (default None)
- **worker_threads** (*None or integer*) – Number of worker threads to use. (default None)

  if None: will use up to 6 threads.

Notes

Outputs csv or txt file with all data and appended taxonomic and gene annotations

'annotated_' + 'denovo_matched' + file

10.2 annotate_denovo

proteoclade.pcannotate.**annotate_denovo**(*file*, *db*, *pctaxa*, *method='dbconstrain'*, *taxon_levels=None*, *worker_threads=None*)

Drives the annotation of denovo/psm-containing files.

Parameters

- **file** (*string*) – csv or txt file containing long-form PSM entries
- **db** (*string*) – PCDB file containing digested peptides to match w/ experiment
- **pctaxa** (*string*) – PCTAXA file containing taxonomic mapping for species and above
- **method** (*string*) – "dbconstrain": serially checks PSM candidates against the PCDB

  "top": only looks at top scoring PSM candidate (default: "dbconstrain")
- **taxon_levels** (*None, string, or tuple*) – Which taxa to annotate above the organism level (default None)
- **worker_threads** (*None or integer*) – Number of worker threads to use. (default None)

  if None: will use up to 6 threads.

Notes

Output is csv or txt file with all data and appended taxonomic and gene annotations

'denovo_matched_' + file

'annotated_' + 'denovo_matched' + file

10.3 filter_taxa

proteoclade.pcannotate.**filter_taxa**(*file*, *taxon_levels*, *taxa*, *unique=False*)

Filters peptide files based on desired taxa.

Parameters

- **file** (*string*) – csv or txt file containing wide-form, peptide entries
- **taxon_levels** (*string, list, or tuple*) – Taxonomic ranks to include in file search. Must be annotated
- **taxa** (*string, list, or tuple*) – Taxa to include in filter
- **unique** (*bool*) – Whether specified taxa must be unique in their given taxonomic rank

Notes

Output is csv or txt file pared down by filter specifications.

'filtered_' + file name

CHAPTER 11

pcquant Module

- *roll_up*

11.1 roll_up

proteoclade.pcquant.**roll_up**(*file*, *unique_taxon='species'*, *inclusion_list=None*, *exclusion_list=None*, *samples='implicit'*, *default_taxon=None*, *missing_values=True*)

Roll up peptides to the gene level, producing gene- and taxon specific outputs.

Parameters

- **file** (*string*) – .csv or tab .txt input file to roll peptides up to genes
- **unique_taxon** (*string*) – Taxon level which is used to determine uniqueness and identity (default "species"). Please check the ncbi_ranks global variable for valid ranks, or use "organisms", after annotating with annotate_peptides or annotate_denovo.
- **inclusion_list** (*None or tuple*) – Taxon members that must be matched for a peptide to be included, if specified (default None)
- **exclusion_list** (*None or tuple*) – Taxon members that must NOT be matched for a peptide to be included, if specified (default None)
- **samples** (*string*) – Method to use for finding samples in file, either "implicit" or "explicit" (default "implicit")
- **default_taxon** (*None or string*) – Member of a taxonomy that will be assigned identity even if the peptide is shared with different members of the same taxonomic level (default None)
- **missing_values** (*bool*) – When adding samples, decide whether to include a gene if any samples have 0 or NaN in their quantitation (default True)

Examples

```
>>> roll_up("annotated_denovo_matched_experiment.txt") # species-specific gene␣
↪rollup
```

Notes

Output is .csv or .txt file with unique taxon, unique genes, and samples

CHAPTER 12

pcconstants Module

- *cleave_rules*
- *uniprot_options*
- *ncbi_ranks*
- *valid_quant_cols*
- *valid_sample_cols*
- *valid_pep_cols*
- *valid_score_cols*
- *basic_annotations*
- *proteoclade_cols*

12.1 cleave_rules

A dictionary that maps a number of proteases to regular expressions.

12.2 uniprot_options

A dictionary that maps UniProt database types to their REST API equivalents.

12.3 ncbi_ranks

A set of all valid taxonomic ranks.

12.4 valid_quant_cols

A tuple of columns ProteoClade will assume are for quantification.

12.5 valid_sample_cols

A tuple of columns ProteoClade will assume are for samples.

12.6 valid_pep_cols

A tuple of columns ProteoClade will assume are for peptide sequences.

12.7 valid_score_cols

A tuple of columns ProteoClade will assume are for scores.

12.8 basic_annotations

A tuple (“organisms”,”genes”) which ProteoClade will annotate by default.

12.9 proteoclade_cols

A set of all columns ProteoClade uses and recognizes.

Python Module Index

p

proteoclade.pcannotate, 22
proteoclade.pcconstants, 26
proteoclade.pcdb, 20
proteoclade.pcquant, 24
proteoclade.pcutilities, 16

Index

A

annotate_denovo() (*in module proteoclade.pcannotate*), 24
annotate_peptides() (*in module proteoclade.pcannotate*), 23

C

create_pcdb() (*in module proteoclade.pcdb*), 21

D

db_fetch_params() (*in module proteoclade.pcutilities*), 19
db_stats() (*in module proteoclade.pcutilities*), 19
download_cRAP() (*in module proteoclade.pcutilities*), 18
download_taxonomy() (*in module proteoclade.pcutilities*), 18
download_uniprot() (*in module proteoclade.pcutilities*), 17
download_uniprot_batch() (*in module proteoclade.pcutilities*), 18

F

filter_taxa() (*in module proteoclade.pcannotate*), 24

L

load_taxonomy() (*in module proteoclade.pcutilities*), 19

M

merge_fastas() (*in module proteoclade.pcdb*), 22

N

ncbi_check() (*in module proteoclade.pcutilities*), 19

P

proteoclade.pcannotate (*module*), 22
proteoclade.pcconstants (*module*), 26
proteoclade.pcdb (*module*), 20
proteoclade.pcquant (*module*), 24
proteoclade.pcutilities (*module*), 16

R

roll_up() (*in module proteoclade.pcquant*), 25