BrEPScmd documentation

Release 1.0.1

Christian-Alexander Dudek
Department of Bioinformatics & Biochemistry
TU Braunschweig

Jun 01, 2017

CONTENTS

Contents:

CHAPTER

ONE

BREPSCMD

1.1 brepscmd.py

The BrEPS command line tool (BrEPScmd) can be used to search a local copy of the database. The tool needs either Python2 or Python 3 to run. For a quick help, please see the online documentation.

This documentation lists all functions used by BrEPScmd and can be used as guide for tool development.

For MySQL usage, mysqlclient is recommended, as it can be used with Python2 and Python3. For Python2 MySQLdb is suitable.

For best performance, Python2 with Googles re2 module is recommended!

class `BrEPScmd.brepscmd.`**`ColumnsAction`**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Argparser action to display usable columns for CSV output.

class `BrEPScmd.brepscmd.`**`Dbase`**(*strType*, *strDB*, *strHost=None*, *strUser=None*, *strPass=None*)

Database class for both MySQL and SQLite usage.

On creation the `strType` can be selected from `MySQL` and `SQLite` to select a database type. All functions are created just like the provided by MySQLdb or sqlite3. MySQL placeholders `%s` are automatically replaced for SQLite. Before every execution the connection is checked to avoid `mysql.OperationalError` exceptions. The returned instance can be directly used for execution. Also the instance needs to be closed properly.

Only non-standard methods are documented!

`strType`
str – Type of database connection (mysql or sqlite).

`strDB`
str – Database name for MySQL, path to file for SQLite.

`strHost`
str – Host for the MySQL connection.

`strUser`
str – Username for the MySQL connection.

`strPass`
str – Password for the MySQL connection.

class `BrEPScmd.brepscmd.`**`LicenseAction`**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Argparser action to display Disclaimer of Warranty.

class `BrEPScmd.brepscmd.`**`WarrantyAction`**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Argparser action to display Disclaimer of Warranty.

BrEPScmd.brepscmd.**complexity**(*strPS*)
Calculates the complexity of a Prosite pattern.

The pattern complexity is calculated for every aminoacid by the following rules:

- A conserved aminoacid has complexity 1/20
- Set of n aminoacids has complexity of n/20
- Wildcard positions (x) have complexity of 1

The sum of all complexities will be divided by the maximum length of the pattern.

Parameters **strPS** (*str*) – Pattern in PROSITE format.

Returns Complexity as float ranged from 1.0 (complex) to 0.0 (not complex).

BrEPScmd.brepscmd.**dna2aa**(*strDNA*)
Converts a DNA sequence to an amino acid sequence.

Parameters **strDNA** (*str*) – Nucleic acid sequence.

Returns The translated amino acid sequence.

Return type str

BrEPScmd.brepscmd.**get_patterns**(*dbCur*, *intMaxLen=7000*, *fltMaxEvalue=1.0*, *lstExclude=[]*)
Gets all sequences with a maximum length of <intMaxLen>.

Parameters

- **dbCur** (Dbase (page 3)) – Database cursor instance of DBase.
- **intMaxLen** (*int*) – The maximum length of a pattern (Default: 7000, basically all patterns).
- **fltMaxEvalue** (*float*) – The maximum E-value of a pattern (Default: 1.0, basically all patterns).
- **lstExclude** (*list*) – A list of pattern types to exclude (Default: [], exclude no pattern types).

Returns List of lists with all patterns for a breps id.

Return type list

BrEPScmd.brepscmd.**parse_fasta**(*strFasta*)
Parses the contents of a FASTA file.

Parameters **strFasta** (*str*) – Contents of a FASTA file.

Returns A dictionary with sequence identifier as key and sequence as value.

Return type dict

BrEPScmd.brepscmd.**prepare_data**(*lstHits*, *dbCur*, *strOrder='evalue'*)
Gets the annotations for a list of BrEPS hits.

Pattern and annotation data is gathered and prepared for output.

Parameters

- **dctHits** (*dict*) – Dictionary with sequence ids as keys and list of pattern ids as values from *search()* (page 5).
- **dbCur** (Dbase (page 3)) – Database cursor instance of DBase.
- **strOrder** (*str*) – Database column name to sort the results.

Returns None.

Raises None.

BrEPScmd.brepscmd.**print_results**(*lstLines*, *lstColumns*, *clsOut*, *strDelimiter*, *strFormat='csv'*)

Prints the data results to terminal or into a file.

Fields containing a delimiter sign will be surrounded by double quotes.

..todo: Add alignment output format!

Parameters

- **lstLines** (*list*) – List of prepared data to display, one for each pattern hit.
- **lstColumns** (*list*) – Columns to display.
- **clsOut** (*instance*) – Class instance to write to. Needs a `write()` method!.
- **strDelimiter** (*str*) – CSV field delimiter.
- **strFormat** (*str*) – Output format, currently only CSV!

BrEPScmd.brepscmd.**ps2re**(*strPS*)

Converts PROSITE pattern to regular expression.

Parameters **strPS** (*str*) – Pattern in PROSITE format.

Returns Pattern in regular expression format.

Return type str

BrEPScmd.brepscmd.**re2ps**(*strRe*)

Converts regular expression pattern to PROSITE.

Parameters **strRe** (*str*) – Pattern in regular expression format.

Returns Pattern in PROSITE format.

Return type str

BrEPScmd.brepscmd.**search**(*lstPatterns*)

Performs the BrEPS search.

Sequences and patterns are given as dictionries in a tuple for multirocessing. The search first checks the standard pattern checks and only checks extended patterns if the standard pattern doesn't hit. The dictionry of sequences resides outside of this function. This is neccessary for multithreading.

Parameters **lstPatterns** (*list*) – List with patterns for one BrEPS id to check.

Returns A dictionry with sequence id as key and a list of pattern ids for every hit.

Return type dict

BrEPScmd.brepscmd.**timeout**(*intSecs*, **args*)

Timeout decorator for the pattern verification.

Complex regular expression search may run for a long time causing the verification to get stuck. The decorator adds a signal that kills a verification that runs longer than `intSecs` seconds.

Parameters **intSecs** (*integer*) – Seconds after which the timeout is fired.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX