prestools Documentation

Release 0.2.1

Roberto Preste

Mar 08, 2022

CONTENTS:

My personal functions and utilities for Python programming.

- Free software: MIT license
- Documentation: https://prestools.readthedocs.io
- GitHub repo: https://github.com/robertopreste/prestools

CHAPTER

ONE

FEATURES

Several ready-to-use functions for different tasks:

- bioinformatics (`prestools.bioinf`)
- data clustering (`prestools.clustering`)
- plotting (`prestools.graph`)
- miscellaneous (`prestools.misc`)

Please refer to the Usage section of the documentation for further details.

CHAPTER

TWO

INSTALLATION

Install prestools using pip (**Python 3 only**):

```
pip install prestools
```

Please refer to the Installation section of the documentation for further details.

CHAPTER

THREE

CREDITS

This package was created with Cookiecutter and the cc-pypackage project template.

CHAPTER

FOUR

TABLE OF CONTENTS

4.1 prestools

My personal functions and utilities for Python programming.

- Free software: MIT license
- Documentation: https://prestools.readthedocs.io
- GitHub repo: https://github.com/robertopreste/prestools

4.1.1 Features

Several ready-to-use functions for different tasks:

- bioinformatics (`prestools.bioinf`)
- data clustering (`prestools.clustering`)
- plotting (`prestools.graph`)
- miscellaneous (`prestools.misc`)

Please refer to the Usage section of the documentation for further details.

4.1.2 Installation

Install prestools using pip **(Python 3 only)**:

```
pip install prestools
```

Please refer to the Installation section of the documentation for further details.

4.1.3 Credits

This package was created with Cookiecutter and the cc-pypackage project template.

4.2 Installation

4.2.1 Stable release

To install prestools, run this command in your terminal:

```
$ pip install prestools
```

This is the preferred method to install prestools, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

4.2.2 From sources

The sources for prestools can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/robertopreste/prestools
```

Or download the tarball:

```
$ curl  -OL https://github.com/robertopreste/prestools/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

4.3 Usage

To use prestools in a project:

```
import prestools
```

`prestools` include several different modules, each one specifically suited for one particular topic or task (bioinformatics, plotting, etc.). It is recommended that you only import the required module, as follows:

```
import prestools.bioinf as pb       # bioinformatics utilities
import prestools.clustering as pc   # clustering utilities
import prestools.graph as pg        # plotting utilities
import prestools.misc as pm         # miscellaneous
```

Some functions are also available as CLI commands, and can be used as follows:

```
prestools bioinf [command] [options]
prestools clustering [command] [options]
prestools misc [command] [options]
```

Please refer to the API page for more information.

4.4 API

All functionalities of prestools can be accessed after having imported the desired module into Python, as in `import prestools.bioinf as pb` and similars. In addition, some functions are also available as Command-Line Interface commands, but this should not be relied on.

4.4.1 Python module functions

prestools.bioinf

prestools.bioinf.**aa_one_to_three**(*sequence: str*) → str
Convert one-letter amino acid code to three-letter code.

Parameters **sequence** – sequence of amino acids in one-letter code

Returns sequence converted to three-letter code

Return type new_seq

prestools.bioinf.**aa_three_to_one**(*sequence: str*) → str
Convert three-letter amino acid code to one-letter code.

Parameters **sequence** – sequence of amino acids in three-letter code

Returns sequence converted to one-letter code

Return type new_seq

prestools.bioinf.**hamming_distance**(*seq_1: str*, *seq_2: str*, *ignore_case: bool = False*) → int
Calculate the Hamming distance between two sequences.

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare
- **ignore_case** – ignore case when comparing sequences (default: False)

Returns Hamming distance

Return type distance

prestools.bioinf.**jukes_cantor_distance**(*seq_1: str*, *seq_2: str*) → float
Calculate the Jukes-Cantor distance between two sequences.

Return the Jukes-Cantor distance between seq_1 and seq_2, calculated as distance = -b log(1 - p/b) where b = 3/4 and p = p_distance.

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare

Returns Jukes-Cantor distance

Return type distance

prestools.bioinf.**kimura_distance**(*seq_1: str*, *seq_2: str*) → float
Calculate the Kimura 2-Parameter distance between two sequences.

Return the Kimura 2-Parameter distance between seq_1 and seq_2, calculated as distance = -0.5 log((1 - 2p -q) * sqrt(1 - 2q)) where p = transition frequency and q = transversion frequency.

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare

Returns Kimura distance

Return type distance

prestools.bioinf.**mutate_sequence**(*sequence: str*, *mutations: int = 1*, *alphabet: str = 'nt'*) → str
Mutate a sequence introducing a given number of mutations.

Introduce a specific number of mutations into the given sequence.

Parameters

- **sequence** – input sequence to mutate
- **mutations** – number of mutations to introduce (default: 1)
- **alphabet** – character alphabet to use ('nt', 'aa') (default: 'nt')

Returns mutated sequence

Return type sequence

prestools.bioinf.**nt_frequency**(*sequence: str*) → Dict[str, float]
Calculate nucleotide frequencies.

Return a dictionary with nucleotide frequencies from the given sequence.

Parameters **sequence** – input nucleotide sequence

Returns dictionary of nucleotide frequencies

Return type freqs

prestools.bioinf.**p_distance**(*seq_1: str*, *seq_2: str*) → float
Calculate the pairwise distance between two sequences.

Return the uncorrected distance between seq_1 and seq_2.

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare

Returns pairwise distance

Return type distance

prestools.bioinf.**quantile_norm**(*x: numpy.ndarray*, *to_log: bool = False*) → numpy.ndarray
Normalize the columns of X to each have the same distribution.

Given an expression matrix (microarray data, read counts, etc) of M genes by N samples, quantile normalization ensures all samples have the same spread of data (by construction).

The data across each row are averaged to obtain an average column. Each column quantile is replaced with the corresponding quantile of the average column.

Parameters

- **x** – array of input data, of shape (N_genes, N_samples)
- **to_log** – log-transform the data before normalising (default: False)

Returns array of normalised data, of shape (N_genes, N_samples)

Return type xn

prestools.bioinf.**random_sequence**(*length: Union[int, str], alphabet: str = 'nt'*) → str
Create a random sequence of the given length.

Create a random sequence of the given length using the specified alphabet (nucleotides or amino acids).

Parameters

- **length** – desired length of the random sequence
- **alphabet** – character alphabet to use ('nt', 'aa') (default: 'nt')

Returns new random sequence

Return type sequence

prestools.bioinf.**reverse_complement**(*sequence: str*, *conversion: str = 'reverse_complement'*) → str
Convert a nucleotide sequence into its reverse complement.

Convert a nucleotide sequence into its reverse, complement or reverse complement.

Parameters

- **sequence** – nucleotide sequence to be converted
- **conversion** – type of conversion to perform ('r'|'reverse', 'c'|'complement', 'rc'|'reverse_complement') (default: 'rc'|'reverse_complement')

Returns converted sequence

prestools.bioinf.**rpkm**(*counts: numpy.ndarray*, *lengths: numpy.ndarray*) → numpy.ndarray
Calculate reads per kilobase transcript per million reads.

RPKM = (10^9 * C) / (N * L)

Where: C = Number of reads mapped to a gene N = Total mapped reads in the experiment L = Exon length in base pairs for a gene

Parameters

- **counts** – count data where columns are individual samples and rows are genes, of shape (N_genes, N_samples)
- **lengths** – gene lengths in base pairs in the same order as the rows in counts, of shape (N_genes,)

Returns

RPKM normalized counts matrix, of shape (N_genes, N_samples)

Return type normed

prestools.bioinf.**shuffle_sequence**(*sequence: str*) → str
Shuffle the given sequence.

Randomly shuffle a sequence, maintaining the same composition.

Parameters **sequence** – input sequence to shuffle

Returns shuffled sequence

Return type tmp_seq

prestools.bioinf.**tajima_nei_distance**(*seq_1: str*, *seq_2: str*) → float
Calculate the Tajima-Nei distance between two sequences.

Return the Tajima-Nei distance between seq_1 and seq_2, calculated as distance = -b log(1 - p / b) where b = 0.5 * [1 - Sum i from A to T(Gi^2+p^2/h)] h = Sum i from A to G(Sum j from C to T (Xij^2/2*Gi*Gj)) p = p-distance Xij = frequency of pair (i,j) in seq1 and seq2, with gaps removed Gi = frequency of base i over seq1 and seq2

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare

Returns Tajima-Nei distance

Return type distance

prestools.bioinf.**tamura_distance**(*seq_1: str*, *seq_2: str*) → float

Calculate the Tamura distance between two sequences.

Return the Tamura distance between seq_1 and seq_2, calculated as distance = -C log(1 - P/C - Q) - 0.5(1 - C)log(1 - 2Q) where P = transition frequency Q = transversion frequency C = GC1 + GC2 - 2 * GC1 * GC2 GC1 = GC-content of seq_1 GC2 = GC-content of seq_2

Parameters

- **seq_1** – first sequence to compare
- **seq_2** – second sequence to compare

Returns Tamura distance

Return type distance

prestools.clustering

prestools.clustering.**find_n_clusters_elbow**(*df: Union[pandas.core.frame.DataFrame, numpy.ndarray], plot: bool = False, method: str = 'ward'*) → Union[int, None, ValueError]

Find the suggested number of clusters using the elbow method.

Find the suggested number of clusters for the given dataframe of correlations, using the elbow method.

Parameters

- **df** – input dataframe of correlations
- **plot** – plot the resulting elbow plot (default: False)
- **method** – method to use to cluster the data ('ward', 'single', 'complete', 'average', 'weighted', 'centroid', 'median') (default: 'ward')

Returns number of clusters found

Return type n_clusters

prestools.clustering.**hierarchical_clustering**(*df: Union[pandas.core.frame.DataFrame, numpy.ndarray], method: str = 'ward'*) → Union[prestools.classes.HierCluster, None, ValueError]

Hierarchical cluster of a dataframe.

Return clustering created using scipy from a given dataframe of correlations, using the HierCluster class available in prestools.classes.

See also:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

Parameters

- **df** – input dataframe of correlations
- **method** – method to use to cluster the data ('ward', 'single', 'complete', 'average', 'weighted', 'centroid', 'median') (default: 'ward')

Returns instance of prestools.classes.HierCluster()

Return type cl

prestools.graph

prestools.graph.**flatten_image**(*img: numpy.ndarray*, *scale: bool = False*) → numpy.ndarray

Convert an image array to a single-dimension vector.

Parameters

- **img** – input image array of shape (l, h, d = 3)
- **scale** – scale resulting vector dividing its values by 255 (default: False)

Returns reshaped vector of shape (l * h * d, 1)

Return type v

prestools.graph.**plot_confusion_matrix**(*cm: numpy.ndarray, class_names: List[str], title: str = 'Confusion Matrix', cmap: str = 'Reds', normalize: bool = False, save: Union[bool, str] = False*)

Create a plot from a confusion matrix array.

Parameters

- **cm** – input confusion matrix array
- **class_names** – class names to use
- **title** – title for resulting plot (default: 'Confusion Matrix')
- **cmap** – colormap to use (default: 'RdBu_r')
- **normalize** – use classes ratios instead of raw numbers (default: False)
- **save** – if False, the plot will not be saved, just shown; otherwise it is possible to specify the path/filename where the file will be saved (default: False)

See also:

http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

prestools.graph.**plot_dendrogram**(*df: Union[pandas.core.frame.DataFrame, numpy.ndarray], cut_off: Union[bool, float] = False, title: str = 'Dendrogram', save: Union[bool, str] = False, method: str = 'ward'*)

Plot a dendrogram plot from a dataframe.

Create (and optionally save) a dendrogram plot starting from a given dataframe of correlations. It is also possible to add a cut-off line given a distance to use for separating clusters.

See also:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

Parameters

- **df** – input dataframe of correlations
- **cut_off** – if not False, a vertical line will be added to better identify clusters (default: False)
- **title** – title for resulting plot (default: 'Dendrogram')
- **save** – if False, the plot will not be saved, just shown; otherwise it is possible to specify the path/filename where the file will be saved (default: False)
- **method** – method to use to cluster the data (default: 'ward')

prestools.graph.**plot_heatmap_dendrogram**(*df: pandas.core.frame.DataFrame*, *cmap: str = 'RdBu_r'*, *title: str = 'Cluster Heatmap'*, *save: Union[bool, str] = False*, *method: str = 'ward'*)

Plot a heatmap with hierarchical clustering of a dataframe.

Create (and optionally save) a heatmap with hierarchical clustering created using Seaborn, starting from a given dataframe of correlations.

See also:

https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

Parameters

- **df** – input dataframe of correlations
- **cmap** – colormap to use (default: 'RdBu_r')
- **title** – title for resulting plot (default: 'Cluster Heatmap')
- **save** – if False, the plot will not be saved, just shown; otherwise it is possible to specify the path/filename where the file will be saved (default: False)
- **method** – method to use to cluster the data (default: 'ward')

prestools.graph.**reduce_xaxis_ticks**(*ax: matplotlib.axes._axes.Axes*, *step: int*)

Show every ith x axis tick.

Parameters

- **ax** – axis to be adjusted
- **step** – factor to reduce the number of x axis ticks by

Examples

```
>>> fig, ax = plt.subplots()
>>> reduce_xaxis_ticks(ax, 5)
```

prestools.graph.**reduce_yaxis_ticks**(*ax: matplotlib.axes._axes.Axes*, *step: int*)

Show every ith y axis tick.

Parameters

- **ax** – axis to be adjusted
- **step** – factor to reduce the number of y axis ticks by

Examples

```
>>> fig, ax = plt.subplots()
>>> reduce_yaxis_ticks(ax, 5)
```

prestools.misc

prestools.misc.**apply_parallel**(*df: pandas.core.frame.DataFrame*, *function: Callable*, *cores: int = 4*) → pandas.core.frame.DataFrame

Apply a function to a dataframe in parallel.

Apply the given function to the dataframe, using the given number of cores for computation. The dataframe will be split in *cores* part, and the function will be applied to each separately; finally, the dataframe is reconstructed and returned.

Parameters

- **df** – input dataframe
- **function** – function to apply
- **cores** – number of cores to use (default: 4)

Returns resulting dataframe

Return type df

prestools.misc.**benchmark**(*function: Callable*) → Callable

Benchmark a given function.

Decorator to run the given function and return the function name and the amount of time spent in executing it.

Parameters **function** – function to benchmark

prestools.misc.**equal_files**(*file1: str*, *file2: str*) → bool

Check whether two files are identical.

First check whether the files have the same size, if so read them and check their content for equality.

Parameters

- **file1** – first file to compare
- **file2** – second file to compare

prestools.misc.**filter_type**(*input_list: List[Any]*, *target_type: Type*) → List[Any]

Only keep elements of a given type from a list of elements.

Traverse a list and return a new list with only elements of the original list belonging to a given type.

Parameters

- **input_list** – input list to filter
- **target_type** – desired type to keep

Returns filtered list

Return type filtered

prestools.misc.**flatten**(*iterable: Iterable*, *drop_null: bool = False*) → List[Any]

Flatten out a nested iterable.

Flatten a nested iterable, even with multiple nesting levels and different data types. It is also possible to drop null values (None) from the resulting list.

Parameters

- **iterable** – nested iterable to flatten
- **drop_null** – filter out None from the flattened list (default: False)

Returns flat list

prestools.misc.**invert_dict**(*input_dict: dict*, *sort_keys: bool = False*) → dict

Create a new dictionary swapping keys and values.

Invert a given dictionary, creating a new dictionary where each key is created from a value of the original dictionary, and its value is the key that it was associated to in the original dictionary (e.g. invert_dict({1: ["A", "E"], 2: ["D", "G"]}) = {"A": 1, "E": 1, "D": 2, "G": 2}). It is also possible to return an inverted dictionary with keys in alphabetical order, although this makes little sense for intrinsically unordered data structures like dictionaries, but it may be useful when printing the results.

Parameters

- **input_dict** – original dictionary to be inverted
- **sort_keys** – sort the keys in the inverted dictionary in alphabetical order (default: False)

Returns inverted dictionary

Return type new_dict

prestools.misc.**prime_factors**(*number: int*) → List[int]

Calculate the prime factors of a number.

Calculate the prime factors of a given natural number. Note that 1 is not a prime number, so it will not be included.

Parameters **number** – input natural number

Returns list of prime factors

Return type factors

prestools.misc.**wordcount**(*sentence: str*, *word: Union[bool, str] = False*, *ignore_case: bool = False*) → Union[dict, int]

Count occurrences of words in a sentence.

Return the number of occurrences of each word in the given sentence, in the form of a dictionary; it is also possible to directly return the number of occurrences of a specific word.

Parameters

- **sentence** – input sentence to count words from
- **word** – target word to count occurrences of
- **ignore_case** – ignore case in the given sentence (default: False)

Returns dictionary of word counts

Return type word_dict

4.4.2 Command Line Interface

prestools bioinf

bioinf

Bioinformatics utilities

```
bioinf [OPTIONS] COMMAND [ARGS]...
```

hamming-distance

Hamming distance between two sequences

Calculate the Hamming distance between SEQ_1 and SEQ_2.

```
bioinf hamming-distance [OPTIONS] SEQ_1 SEQ_2
```

Options

-i, --ignore_case
Ignore case when comparing sequences (default: False)

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

jukes-cantor-distance

Jukes-Cantor distance between two sequences

Return the Jukes-Cantor distance between SEQ_1 and SEQ_2, calculated as distance = -b log(1 - p/b) where b = 3/4 and p = p_distance.

```
bioinf jukes-cantor-distance [OPTIONS] SEQ_1 SEQ_2
```

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

kimura-distance

Kimura 2-Parameter distance between two sequences

Return the Kimura 2-Parameter distance between SEQ_1 and SEQ_2, calculated as distance = -0.5 log((1 - 2p -q) * sqrt(1 - 2q)) where p = transition frequency and q = transversion frequency.

```
bioinf kimura-distance [OPTIONS] SEQ_1 SEQ_2
```

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

p-distance

Pairwise distance between two sequences

Return the uncorrected distance between SEQ_1 and SEQ_2.

```
bioinf p-distance [OPTIONS] SEQ_1 SEQ_2
```

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

random-sequence

Create a random sequence of the given length

Create a random sequence of the given LENGTH using the specified ALPHABET (nucleotides or aminoacids).

```
bioinf random-sequence [OPTIONS] LENGTH
```

Options

-a, --alphabet <alphabet>
Character alphabet to use to create the sequence ('nt', 'aa') (default: 'nt')

Options ntlaa

Arguments

LENGTH
Required argument

reverse-complement

Convert a nucleotide sequence into its reverse complement

Convert a nucleotide SEQUENCE into its reverse, complement or reverse complement.

```
bioinf reverse-complement [OPTIONS] SEQUENCE
```

Options

-c, --conversion <conversion>
Type of conversion to perform ('r'|'reverse', 'c'|'complement', 'rc'|'reverse_complement') (default: 'rc'|'reverse_complement')

Options reverse|complement|reverse_complement|r|c|rc

Arguments

SEQUENCE
Required argument

shuffle-sequence

Shuffle the given sequence

Randomly shuffle a SEQUENCE, maintaining the same nucleotide composition.

```
bioinf shuffle-sequence [OPTIONS] SEQUENCE
```

Arguments

SEQUENCE
Required argument

tajima-nei-distance

Tajima-Nei distance between two sequences

Return the Tajima-Nei distance between SEQ_1 and SEQ_2, calculated as distance = -b log(1 - p / b) where b = 0.5 * [1 - Sum i from A to T(Gi^2+p^2/h)] h = Sum i from A to G(Sum j from C to T (Xij^2/2*Gi*Gj)) p = p-distance Xij = frequency of pair (i,j) in SEQ_1 and SEQ_2, with gaps removed Gi = frequency of base i over SEQ_1 and SEQ_2

```
bioinf tajima-nei-distance [OPTIONS] SEQ_1 SEQ_2
```

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

tamura-distance

Tamura distance between two sequences

Return the Tamura distance between SEQ_1 and SEQ_2, calculated as distance = -C log(1 - P/C - Q) - 0.5(1 - C)log(1 - 2Q) where P = transition frequency Q = transversion frequency C = GC1 + GC2 - 2 * GC1 * GC2 GC1 = GC-content of SEQ_1 GC2 = GC-content of SEQ_2

```
bioinf tamura-distance [OPTIONS] SEQ_1 SEQ_2
```

Arguments

SEQ_1
Required argument

SEQ_2
Required argument

prestools clustering

clustering

Data clustering utilities

```
clustering [OPTIONS] COMMAND [ARGS]...
```

find-n-clusters-elbow

Find the number of clusters using the elbow method

Find the suggested number of clusters for the given dataframe of correlations, using the elbow method.

```
clustering find-n-clusters-elbow [OPTIONS] DF
```

Options

-m, --method <method>
Method to be used to cluster the data [‘ward’, ‘single’, ‘complete’, ‘average’, ‘weighted’, ‘centroid’, ‘median’] (default = ‘ward’)

Options ward|single|complete|average|weighted|centroid|median

Arguments

DF
Required argument

prestools misc

misc

Miscellaneous utilities

```
misc [OPTIONS] COMMAND [ARGS]...
```

equal-files

Check whether two files are identical

First check whether FILE1 and FILE2 have the same size, if so read them and check their content for equality.

```
misc equal-files [OPTIONS] FILE1 FILE2
```

Arguments

FILE1
Required argument

FILE2
Required argument

prime-factors

Calculate the prime factors of a number

Calculate the prime factors of a given natural NUMBER. Note that 1 is not a prime number, so it will not be included.

```
misc prime-factors [OPTIONS] NUMBER
```

Arguments

NUMBER
Required argument

wordcount

Count occurrences of words in a sentence

Return the number of occurrences of each word in the given SENTENCE, in the form of a dictionary; it is also possible to directly return the number of occurrences of a specific WORD.

```
misc wordcount [OPTIONS] SENTENCE
```

Options

-w, --word <word>
Target word to count occurrences of

-i, --ignore_case
Ignore case in the given sentence (default: False)

Arguments

SENTENCE
Required argument

4.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/robertopreste/prestools/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

prestools could always use more documentation, whether as part of the official prestools docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/robertopreste/prestools/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.5.2 Get Started!

Ready to contribute? Here's how to set up *prestools* for local development.

1. Fork the *prestools* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/prestools.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv prestools
$ cd prestools/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 prestools tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/robertopreste/prestools/pull_requests and make sure that the tests pass for all supported Python versions.

4.5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_prestools
```

4.5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

4.6 Credits

4.6.1 Development Lead

- Roberto Preste <robertopreste@gmail.com>

4.6.2 Contributors

None yet. Why not be the first?

4.7 History

4.7.1 0.1.0 (2019-03-07)

- First development release.

0.1.1 (2019-03-14)

- Add functions for clustering data.

0.1.2 (2019-03-15)

- Add `hamming_distance()` and `mutate_sequence()` functions in `prestools.bioinf` and related tests;
- Clean code style.

0.1.3 (2019-03-19)

- Add command line interface commands and related tests for `bioinf` and `misc` modules;
- Add `filter_type()` function in `prestools.misc` and related tests;
- Clean code style.

0.1.4 (2019-03-23)

- Add `wordcount()` function in `prestools.misc` and related tests.

0.1.5 (2019-04-05)

- Add `equal_files()` function in `prestools.misc` and related tests;
- Update docstrings.

0.1.6 (2019-04-11)

- Add `random_image()` function and CLI in `prestools.plotting`.

0.1.7 (2019-04-20)

- Add `benchmark()` function in `prestools.misc` and related tests.

0.1.8 (2019-04-26)

- Change `pm.benchmark()` function to a decorator;
- Add several distance calculation functions to `prestools.bioinf` and related tests;
- Reformat code in `prestools.bioinf`.

0.1.9 (2019-04-27)

- Add distance functions to `bioinf` CLI command and related tests.

0.1.10 (2019-05-06)

- Change `plotting` library name to `graph` (to avoid alias conflict with pandas_profiling).

0.1.11 (2019-05-07)

- Fix docstrings and type hints;
- Update documentation.

0.1.12 (2019-05-08)

- Add `apply_parallel` function to `prestools.misc`.

0.1.13 (2019-06-22)

- Add short version arguments to `reverse_complement` bioinf function.

0.1.14 (2019-07-03)

- Add `learn` module and related tests;
- Remove `random_image` from `graph` module;
- Clean code.

4.7.2 0.2.0 (2019-08-14)

- Add `rpkm` and `quantile_norm` functions to `prestools.bioinf`;
- Add `reduce_xaxis_ticks` and `reduce_yaxis_ticks` functions to `prestools.graph`;
- Move `flatten_image` to `prestools.graph`;
- Remove `prestools.learn`;
- Update related tests;
- Fix documentation and API.

0.2.1 (2019-09-07)

- Add `plot_confusion_matrix` to `prestools.graph`;
- Add related tests;
- Drop support for Python < 3.6;
- Update requirements;
- Update documentation.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

A

B

C

D

E

F

T

W