yabul

Release 0.0.3

Tim O'Donnell

Mar 04, 2021

CONTENTS

CHAPTER

ONE

YABUL

Yet Another Bioinformatics Utility Library

1.1 Submodules

1.1.1 yabul.align

1.1.2 yabul.fasta

FASTA reading and writing

1.2 Functions

- *read_fasta()*: Parse a fasta file to a pandas DataFrame.
- *write_fasta()*: Write sequences to a FASTA.
- *align_pair()*: Align two protein or DNA sequences.

yabul.**read_fasta**(*filename*)

Parse a fasta file to a pandas DataFrame.

Compression is supported (via pandas read_csv) and is inferred by extension: '.gz', '.bz2', '.zip', or '.xz'.

Parameters **filename** (*string*) –

Returns

- *pandas.DataFrame with columns "description" and "sequence". The index of the*
- *DataFrame is the "sequence ID", i.e. the first space-separated token of the*
- *description.*

yabul.**write_fasta**(*filename*, *sequences*)

Write sequences to a FASTA.

Parameters

- **filename** (*string*) – File to write. If it ends with '.gz' the file will be gzip compressed.
- **sequences** (*iterable of (name, sequence) pairs*) – Sequences to write. Both name and sequence should be strings.

yabul.**align_pair**(*query_seq*, *reference_seq*, *local=False*, *gap_open_penalty=11*, *gap_extension_penality=1*, *substitution_matrix='blosum62'*, *alignment_function=None*)

Align two protein or DNA sequences.

By default, a protein substitution matrix (blosum62) is used. If you are aligning DNA or RNA, you should use a nucleotide substitution matrix by passing, for example, substitution_matrix="dnafull".

This is a thin wrapper over the Parasail library implementation.

Returns a pandas.Series with the results of the alignment.

Parameters

- **query_seq** (*string*) – First sequence to align
- **reference_seq** (*string*) – Second sequence to align.
- **local** (*boolean*) – If True, a local alignment is performed using the Smith-Waterman algorithm. This means that gaps at the beginning or end of the sequences are not penalized, and only the part of the sequences that align are returned.

  If False, a global alignment is performed using the Needleman-Wunsch algorithm. This means that the two sequences will be aligned in their entirety.
- **gap_open_penalty** (*int*) – Penality for starting a gap
- **gap_extension_penality** (*int*) – Penalty for extending a gap
- **substitution_matrix** (*string*) – Name of substitution matrix. Examples: "blosum62", "blosum90", "dnafull", "pam100". If you are aligning DNA or RNA you should use a nucleotide substitution matrix, such as "dnafull".

  Full list of supported matrices: https://github.com/jeffdaily/parasail/tree/master/parasail/matrices
- **alignment_function** (*function*) – Advanced use. If you know the underlying parasail alignment function you would like to use, you can pass it here. Otherwise a reasonable default is used.

Returns

query [string] Aligned query sequence

reference [string] Aligned reference sequence

correspondence [string] Characters (similar to BLAST "midline") indicating the correspondence between query and reference strings.

score [int] Alignment score. Higher indicates a better alignment.

Return type pandas.Series with keys

CHAPTER

TWO

YABUL

Yet Another Bioinformatics Utilities Library

This is a small collection of Python functions for working with protein, DNA, and RNA sequences. We use pandas data frames wherever possible.

Yabul currently supports:

- Reading and writing FASTAs
- Pairwise local and global sequence alignment (uses parasail)

Requires Python 3.6+.

2.1 Installation

Install using pip:

```
$ pip install yabul
```

You can run the unit from a checkout of the repo as follows:

```
$ pip install pytest
$ pytest
```

2.2 Example

2.2.1 Reading and writing FASTAs

The read_fasta function returns a ``*pandas.DataFrame*` <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>`_:

```
>>> import yabul
>>> df = yabul.read_fasta("test/data/cov2.fasta")
>>> df.head(3)
                                                                description␣
↪                               sequence
id
sp|P0DTC2|SPIKE_SARS2  sp|P0DTC2|SPIKE_SARS2 Spike glycoprotein OS=Se... ␣
↪MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSS...
```

(continues on next page)

(continued from previous page)

```
sp|P0DTD1|R1AB_SARS2   sp|P0DTD1|R1AB_SARS2 Replicase polyprotein 1ab... ␣
↪MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHL...
sp|P0DTC1|R1A_SARS2    sp|P0DTC1|R1A_SARS2 Replicase polyprotein 1a O... ␣
↪MESLVPGFNEKTHVQLSLPVLQVRDVLVRGFGDSVEEVLSEARQHL...
```

The write_fasta function takes (name, sequence) pairs:

```
>>> yabul.write_fasta("out.fasta", [("protein1", "TEST"), ("protein2", "HIHI")])
>>> yabul.write_fasta("out2.fasta", df.sequence.items())
```

2.2.2 Sequence alignment

The align_pair function will give a local (Smith-Waterman) and global (Needleman-Wunsch) alignment of two sequences. It returns a pandas.Series with the aligned sequences.

By default, the alignment is global:

```
>>> yabul.align_pair("AATESTDD", "TEST")
query             AATESTDD
reference         --TEST--
correspondence      ||||
score                  -5
dtype: object
```

To do a local alignment, pass `local=True`.

```
>>> yabul.align_pair("AATESTDD", "TEST", local=True)
query             TEST
reference         TEST
correspondence    ||||
score               19
dtype: object
```

2.3 Dependencies

The alignment routine is a thin wrapper around the Smith-Waterman and Needleman-Wunsch implementations from parasail.

2.4 Contributing

We welcome contributions of well-documented code to read and write common bioinformatics file formats using pandas objects. Please include unit tests in your PR. Additional functionality like multiple sequence alignment would also be nice to add.

2.5 Releasing

To push a new release to PyPI:

- Make sure the package version specified in ``__init__.py` <https://github.com/timodonnell/yabul/blob/main/yabul/__init__.py>`_ is a new version greater than what's on PyPI.
- Tag a new release on GitHub matching this version

Travis should deploy the release to PyPI automatically.

Documentation at https://yabul.readthedocs.io/en/latest/ should update automatically on commit.

To build the documentation locally, run:

```
$ cd docs
$ pip install -r requirements.txt
$ sphinx-build -b html . _build
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

y

INDEX