seqm Documentation

Release 0.1.0

Blake Printy

Mar 18, 2021

CONTENTS

CHAPTER

ONE

OVERVIEW

The seqm module contains functions for calculating sequence-related distance and complexity metrics, commonly used in language processing and next-generation sequencing. It has a simple and consistent API that be used for investigating sequence characteristics:

```
>>> import seqm
>>> seqm.hamming('ATTATT', 'ATTAGT')
1
>>> seqm.edit('ATTATT', 'ATAGT')
2
>>> seqm.polydict('AAAACCGT')
{'A': 4, 'C': 2, 'G': 1, 'T': 1}
>>> seqm.polylength('AAAACCGT')
4
>>> seqm.entropy('AGGATAAG')
1.40
>>> seqm.gc_percent('AGGATAAG')
0.375
>>> seqm.gc_skew('AGGATAAG')
3.0
>>> seqm.gc_shift('AGGATAAG')
1.67
>>> seqm.dna_weight('AGGATAAG')
3968.59
>>> seqm.rna_weight('AGGATAAG')
4082.59
>>> seqm.aa_weight('AGGATAAG')
700.8
>>> seqm.tm('AGGATAAGAGATAGATTT')
39.31
>>> seqm.zipsize('AGGATAAGAGATAGATTT')
22
```

For more information on available functionality, see the Usage section of the documentation.

CHAPTER

TWO

CONTENT

2.1 Installation

2.1.1 pip

To install the latest stable version of the project from pip use:

```
$ pip install seqm
```

2.1.2 Source

To install the latest version of the project from source, close the repository from GitHub and use setuptools:

```
$ git clone http://github.com/atgtag/seqm.git
$ cd seqm
$ python setup.py install
```

2.1.3 Questions/Feedback

File an issue in the GitHub issue tracker.

2.2 Usage

2.2.1 Overview

The seqm module contains functions for calculating sequence-related distance and complexity metrics, commonly used in language processing and next-generation sequencing. It has a simple and consistent API that be used for investigating sequence characteristics:

```
>>> import seqm
>>> seqm.hamming('ATTATT', 'ATTAGT')
1
>>> seqm.edit('ATTATT', 'ATAGT')
2
>>> seqm.polydict('AAAACCGT')
{'A': 4, 'C': 2, 'G': 1, 'T': 1}
>>> seqm.polylength('AAAACCGT')
```

(continues on next page)

(continued from previous page)

```
4
>>> seqm.entropy('AGGATAAG')
1.40
>>> seqm.gc_percent('AGGATAAG')
0.375
>>> seqm.gc_skew('AGGATAAG')
3.0
>>> seqm.gc_shift('AGGATAAG')
1.67
>>> seqm.dna_weight('AGGATAAG')
3968.59
>>> seqm.rna_weight('AGGATAAG')
4082.59
>>> seqm.aa_weight('AGGATAAG')
700.8
>>> seqm.zipsize('AGGATAAGAGATAGATTT')
22
```

It also has a *seqm.Sequence* object for object-based access to these properties:

```
>>> import seqm
>>> seq = seqm.Sequence('AAAACCGT')
>>> seq.hamming('AAAAGCGT')
1
>>> seq.gc_percent
0.375
>>> seq.revcomplement
ACGTACGT
>>> seq.dna_weight
3895.59
```

All of the metrics available in the repository are listed below, and can also be found in the API section of the documentation.

2.2.2 List of Available Functions

Sequence Quantification

Function	Metric
polydict()	Length of longest homopolymer for all bases in sequence.
polylength()	Length of longest homopolymer in sequence.
entropy()	Shannon entropy for bases in sequence.
gc_percent()	Percentage of GC bases in sequence relative to all bases.
gc_skew()	GC skew for sequence: (#G - #C)/(#G + #C).
gc_shift()	GC shift for sequence: (#A + #T)/(#G + #C)
dna_weight()	Molecular weight for sequence with DNA backbone.
rna_weight()	Molecular weight for sequence with RNA backbone.
aa_weight()	Molecular weight for amino acid sequence.
zipsize()	Compressibility of sequence.
tm()	Melting temperature of sequence.

Domain Conversion

Function	Conversion
revcomplement()	Length of longest homopolymer for all bases in sequence.
complement()	Length of longest homopolymer in sequence.
aa()	Shannon entropy for bases in sequence.
wrap()	Percentage of GC bases in sequence relative to all bases.
likelihood()	GC skew for sequence: (#G - #C)/(#G + #C).
qscore()	GC shift for sequence: (#A + #T)/(#G + #C)

Distance Metrics

Function	Distance Metric
hamming()	Hamming distance between sequences.
edit()	Edit (levenshtein) distance between sequences

Utilities

Function	Utility
`random_sequence()`	Generate random sequence.
wrap()	Newline-wrap sequence

2.2.3 Command-Line Usage

Once seqm is installed, all methods can be accessed via the `seqm` entry point:

```
~$ seqm
```

To run a specific method on a sequence, use:

```
~$ seqm gc_skew AGTAGTAGTTTAGGTTAGGTAG
8.0
```

For commands comparing sequences, simply use both sequences as arguments:

```
~$ seqm edit AGTAGTAGTAGTAT AGTAGTAGTAGAAAAT
3
```

And finally, to supply command line arguments to a method, do the following:

```
~$ seqm wrap --bases=10 AGTAGTAGTAGTATAGTAGTAGTAGAAAAT
AGTAGTAGTA
GTATAGTAGT
AGTAGAAAAT
```

You can also pipe commands with the cli tool:

```
~$ seqm random --length 10 | seqm wrap --bases 5 -
ATGGA
TATTA
```

2.3 API

2.3.1 Sequence Composition Metrics

seqm.**polydict**(*seq*, *nuc='ACGT'*)

Computes largest homopolymer for all specified nucleotides.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.polydict('AAAACCGT')
{'A': 4, 'C': 2, 'G': 1, 'T': 1}
```

seqm.**polylength**(*seq*)

Calculate length of maximum homopolymer stretch within sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.polylength('AAAACCGT')
4
```

seqm.**entropy**(*seq*)

Calculate Shannon entropy of sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.entropy('AGGATAAG')
1.40
>>> sequtils.entropy('AAAACCGT')
1.75
```

seqm.**gc_percent**(*seq*)

Calculate fraction of GC bases within sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.gc_percent('AGGATAAG')
0.375
```

seqm.**gc_skew**(*seq*)

Calculate GC skew (g-c)/(g+c) for sequence. For homopolymer stretches with no GC, the skew will be rounded to zero.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.gc_skew('AGGATAAG')
3.0
```

seqm.**gc_shift**(*seq*)

Calculate GC shift (a + t)/(g + c) for sequence. For homopolymer stretches with no GC, the shift will be rounded to the number of bases in the sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.gc_shift('AGGATAAG')
1.67
```

seqm.**dna_weight**(*seq*)

Return molecular weight of triphosphate dna sequence (g/mol).

See https://www.thermofisher.com/us/en/home/references/ambion-tech-support/rna-tools-and-calculators/dna-and-rna-molecular-weights-and-conversions.html for details on conversions.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.dna_weight('AGGATAAG')
3968.59
```

seqm.**rna_weight**(*seq*)

Return molecular weight of triphosphate rna sequence (g/mol).

See https://www.thermofisher.com/us/en/home/references/ambion-tech-support/rna-tools-and-calculators/dna-and-rna-molecular-weights-and-conversions.html for details on conversions.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.rna_weight('AGGATAAG')
4082.59
```

seqm.**aa_weight**(*seq*)

Return molecular weight of amino acid sequence (g/mol).

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.aa_weight('AGGATAAG')
700.8
```

seqm.**zipsize**(*seq*)
Calculate size of gzip-compressed sequence.

Parameters **seq** (*str*) – Sequence

Examples

```
>>> sequtils.zipsize('AGGATAAGAGATAGATTT')
22
```

seqm.**tm**(*seq*, *mv=50*, *dv=1.5*, *n=0.6*, *d=50*, *tp=1*, *sc=1*)
Calculate size of gzip-compressed sequence.

Parameters

- **seq** (*str*) – Sequence
- **mv** (*float*) – Concentration of monovalent cations in mM, by default 50mM
- **dv** (*float*) – Concentration of divalent cations in mM, by default 1.5mM
- **n** (*float*) – Concentration of deoxynycleotide triphosphate in mM, by default 0.6mM
- **d** (*float*) – Concentration of DNA strands in nM, by default 50nM
- **tp** (*int*) – Specifies the table of thermodynamic parameters and the method of melting temperature calculation (default 1):

  0 Breslauer et al., 1986 and Rychlik et al., 1990 (used by primer3 up to and including release 1.1.0).

  1 Use nearest neighbor parameters from SantaLucia 1998
- **sc** (*int*) – Specifies salt correction formula for the melting temperature calculation (default 1):

  0 Schildkraut and Lifson 1965, used by primer3 up to and including release 1.1.0.

  1 SantaLucia 1998 2 Owczarzy et al., 2004

Examples

```
>>> sequtils.tm('AGGATAAGAGATAGATTT')
39.31
```

2.3.2 Domain Conversion

seqm.**revcomplement**(*seq*)
Reverse complement sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.revcomplement('AACCTT')
'AAGGTT'
```

seqm.**complement**(*seq*)
Complement sequence.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.complement('AACCTT')
TTGGAA
```

seqm.**aa**(*seq*)
Return amino acid translation of sequence. Ends of the sequences that don't produce a full codon will be clipped.

Parameters **seq** (*str*) – Nucleotide sequence

Examples

```
>>> sequtils.aa('ATGTAG')
M*
```

seqm.**likelihood**(*seq*)
Translates quality scores sequence into error likelihoods.

Parameters **seq** (*str*) – Sequence of quality scores.

seqm.**qscore**(*seq*)
Translates quality score sequence into phred-scaled likelihoods.

Parameters **seq** (*str*) – Sequence of quality scores.

2.3.3 Sequence Similarity Metrics

seqm.**hamming**(*seq1*, *seq2*)
Calculate hamming distance between sequences.

Parameters

- **seq1** (*str*) – Reference sequence
- **seq2** (*str*) – Sequence to compare

Examples

```
>>> hamming('AACCTT', 'AAGCCTT')
1
```

seqm.**edit**(*seq1*, *seq2*)
Wrapper around editdistance.eval for fast Levenshtein distance computation.

Parameters

- **seq1** (*str*) – Reference sequence
- **seq2** (*str*) – Sequence to compare

Examples

```
>>> edit('banana', 'bahama')
2
```

2.3.4 Sequence-Related Utilities

seqm.**random**(*length=11*, *template='ACGT'*)
Generate random sequence of specified length.

Parameters

- **length** (*int*) – Length of random sequence to generate.
- **template** (*str*) – String with bases to use in generating sequence.

Example

```
>>> sequtils.random_sequence()
CGGACGGTATG
>>> sequtils.random_sequence(length=5, template='AC')
CCCAA
```

seqm.**wrap**(*seq*, *bases=60*)
Print wrapped sequence.

Parameters

- **seq** (*str*) – Nucleotide sequence
- **bases** (*int*) – Number of bases to include on each line.

Example

```
>>> sequtils.wrap(CGGACGGTATG, bases=3)
CGG
ACG
GTA
TG
```

2.3.5 Objects

class seqm.**Sequence**(*sequence*)

Object for managing sequence structure and operating on sequences (i.e. getting amino acid sequence, reverse complement, gc content, etc . . .).

Parameters **sequence** (*str*) – Nucleotide sequence.

Examples

```
>>> seq = sequtils.Sequence('ACGTACGT')
>>> seq.gc_content
0.25
>>> seq.revcomplement
ACGTACGT
>>> seq.dna_weight
3895.59
```

aa

Wrapper around `sequtils.aa()` for the `sequtils.Sequence` object.

aa_weight

Wrapper around `sequtils.aa_weight()` for the `sequtils.Sequence` object.

complement

Wrapper around `sequtils.complement()` for the `sequtils.Sequence` object.

dna_weight

Wrapper around `sequtils.dna_weight()` for the `sequtils.Sequence` object.

edit(*other*)

Wrapper around `sequtils.edit()` for the `sequtils.Sequence` object.

Parameters **other** (*str, Sequence*) – Sequence to compare.

entropy

Wrapper around `sequtils.entropy()` for the `sequtils.Sequence` object.

gc_percent

Wrapper around `sequtils.gc_percent()` for the `sequtils.Sequence` object.

gc_shift

Wrapper around `sequtils.gc_shift()` for the `sequtils.Sequence` object.

gc_skew

Wrapper around `sequtils.gc_skew()` for the `sequtils.Sequence` object.

hamming(*other*)

Wrapper around `sequtils.hamming()` for the `sequtils.Sequence` object.

Parameters **other** (*str, Sequence*) – Sequence to compare.

polydict
Wrapper around `sequtils.polydict()` for the `sequtils.Sequence` object.

polylength
Wrapper around `sequtils.polylength()` for the `sequtils.Sequence` object.

revcomplement
Wrapper around `sequtils.revcomplement()` for the `sequtils.Sequence` object.

rna_weight
Wrapper around `sequtils.rna_weight()` for the `sequtils.Sequence` object.

tm
Wrapper around `sequtils.zipsize()` for the `sequtils.Sequence` object.

wrap (*bases=60*)
Wrapper around `sequtils.wrap()` for the `sequtils.Sequence` object.

Parameters **bases** (*int*) – Number of bases to include in line.

zipsize
Wrapper around `sequtils.zipsize()` for the `sequtils.Sequence` object.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX