pyfastx

Release 0.9.0

Lianming Du

Nov 30, 2022

CONTENTS:

1 Installation **3**
- 1.1 Install from PyPI . . . 3
- 1.2 Install from source . . . 3

2 FASTX **5**
- 2.1 Iterate over sequences in FASTA . . . 5
- 2.2 Iterate over reads in FASTQ . . . 5

3 FASTA **7**
- 3.1 Read FASTA file . . . 7
- 3.2 FASTA records iteration . . . 7
- 3.3 Get FASTA information . . . 8
- 3.4 Get longest and shortest sequence . . . 8
- 3.5 Calculate N50 and L50 . . . 9
- 3.6 Get sequence mean and median length . . . 9
- 3.7 Get sequence counts . . . 10
- 3.8 Get subsequences . . . 10
- 3.9 Get flank sequences . . . 10
- 3.10 Key function . . . 11

4 Sequence **13**
- 4.1 Get a sequence from FASTA . . . 13
- 4.2 Get sequence information . . . 13
- 4.3 Sequence slice . . . 14
- 4.4 Reverse and complement sequence . . . 15
- 4.5 Read sequence line by line . . . 15
- 4.6 Search for subsequence . . . 16

5 FASTQ **17**
- 5.1 Read FASTQ file . . . 17
- 5.2 FASTQ records iteration . . . 17
- 5.3 Get FASTQ information . . . 18

6 Read **19**
- 6.1 Get read from FASTQ . . . 19
- 6.2 Get read information . . . 19

7 FastaKeys **21**
- 7.1 Get fasta keys . . . 21
- 7.2 Sort keys . . . 21
- 7.3 Filter keys . . . 22

7.4 Clear filter and sort order . . . 23

8 FastqKeys **25**
8.1 Get fastq keys . . . 25

9 Command line interface **27**
9.1 Build index . . . 27
9.2 Show statistics information . . . 28
9.3 Split FASTA/Q file . . . 29
9.4 Convert FASTQ to FASTA file . . . 29
9.5 Get subsequence with region . . . 29
9.6 Sample sequences . . . 30
9.7 Extract sequences . . . 30

10 Multiple processes **33**
10.1 Example one . . . 33

11 Drawbacks **35**

12 Changelog **37**
12.1 Version 0.8.4 (2021-06-30) . . . 37
12.2 Version 0.8.3 (2021-04-25) . . . 37
12.3 Version 0.8.2 (2021-01-02) . . . 37
12.4 Version 0.8.1 (2020-12-16) . . . 37
12.5 Version 0.8.0 (2020-12-15) . . . 38
12.6 Version 0.7.0 (2020-09-20) . . . 38
12.7 Older versions . . . 38

13 API Reference **49**
13.1 pyfastx.version . . . 49
13.2 pyfastx.Fasta . . . 49
13.3 pyfastx.Sequence . . . 52
13.4 pyfastx.Fastq . . . 53
13.5 pyfastx.Read . . . 55
13.6 pyfastx.Fastx . . . 55
13.7 pyfastx.FastaKeys . . . 56
13.8 pyfastx.FastqKeys . . . 56

14 Acknowledgements **57**

15 Indices and tables **59**

Index **61**

The `pyfastx` is a lightweight Python C extension that enables users to randomly access to sequences from plain and **gzipped** FASTA/Q files. This module aims to provide simple APIs for users to extract sequence from FASTA and reads from FASTQ by identifier and index number. The `pyfastx` will build indexes stored in a sqlite3 database file for random access to avoid consuming excessive amount of memory. In addition, the `pyfastx` can parse standard (*sequence is spread into multiple lines with same length*) and nonstandard (*sequence is spread into one or more lines with different length*) FASTA format. This module used kseq.h written by @attractivechaos in klib project to parse plain FASTA/Q file and zran.c written by @pauldmccarthy in project indexed_gzip to index gzipped file for random access.

This project was heavily inspired by @mdshw5's project pyfaidx and @brentp's project pyfasta.

Features

- Single file for the Python extension
- Lightweight, memory efficient for parsing FASTA file
- Fast random access to sequences from **gzipped** FASTA file
- Read sequences from FASTA file line by line
- Calculate assembly N50 and L50
- Calculate GC content and nucleotides composition
- Extract reverse, complement and antisense sequence
- Excellent compatibility, support for parsing nonstandard FASTA file
- Support for random access reads from FASTQ file

CHAPTER

ONE

INSTALLATION

You can install pyfastx via the Python Package Index (PyPI) (**recommended**) or from source.

Make sure you have installed both pip and Python before starting.

Currently, `pyfastx` supports Python 3.5, 3.6, 3.7, 3.8, 3.9 and can work on Windows, Linux, MacOS.

> **Warning:** Python 2.7 has been retired. Python 3.4 has now reached its end-of-life and has been retired

1.1 Install from PyPI

```
pip install pyfastx
```

Update pyfastx using `pip`

```
pip install -U pyfastx
```

1.2 Install from source

`pyfastx` depends on zlib and sqlite3. If you want to compile and install pyfastx from source code. First, you should install zlib and sqlite3.

On Centos

```
yum install zlib-devel
yum install sqlite-devel
```

On Ubuntu

```
apt install zlib1g-dev
apt install libsqlite3-dev
```

On MacOS

```
brew install zlib
brew install sqlite3
```

Second, clone pyfastx using `git` or download latest release:

```
git clone https://github.com/lmdu/pyfastx.git
```

Then, cd to the pyfastx folder and run install command:

```
cd pyfastx
python setup.py install
```

CHAPTER

TWO

FASTX

New in pyfastx 0.8.0.

2.1 Iterate over sequences in FASTA

When iterating over sequences on FASTX object, a tuple `(name, seq, comment)` will be returned, the comment is the content of header line after the first white space character.

```
>>> fa = pyfastx.Fastx('tests/data/test.fa')
>>> for name,seq,comment in fa:
>>>     print(name)
>>>     print(seq)
>>>     print(comment)

>>> #always output uppercase sequence
>>> for item in pyfastx.Fastx('tests/data/test.fa', uppercase=True):
>>>     print(item)

>>> #Manually specify sequence format
>>> for item in pyfastx.Fastx('tests/data/test.fa', format="fasta"):
>>>     print(item)
```

2.2 Iterate over reads in FASTQ

When iterating over reads on FASTX object, a tuple `(name, seq, qual, comment)` will be returned, the comment is the content of header line after the first white space character.

```
>>> fq = pyfastx.Fastx('tests/data/test.fq')
>>> for name,seq,qual,comment in fq:
>>>     print(name)
>>>     print(seq)
>>>     print(qual)
>>>     print(comment)
```

CHAPTER

THREE

FASTA

3.1 Read FASTA file

Read plain or gzipped FASTA file and build index, support for random access to FASTA.

```
>>> import pyfastx
>>> fa = pyfastx.Fasta('test/data/test.fa.gz')
>>> fa
<Fasta> test/data/test.fa.gz contains 211 seqs
```

Note: Building index may take some time. The time required to build index depends on the size of FASTA file. If index built, you can randomly access to any sequences in FASTA file. The index file can be reused to save time when you read sequences from FASTA file next time.

3.2 FASTA records iteration

The fastest way to iterate plain or gzipped FASTA file without building index, the iteration will return a tuple contains name and sequence.

```
>>> import pyfastx
>>> for name, seq in pyfastx.Fasta('test/data/test.fa.gz', build_index=False):
>>>     print(name, seq)
```

If you want to use full header line as sequence identifier without building index, you can do like this:

```
>>> import pyfastx
>>> for name, seq in pyfastx.Fasta('test/data/test.fa', build_index=False, full_
↪name=True):
>>>     print(name, seq)
```

You can also iterate sequence object from FASTA object like this:

```
>>> import pyfastx
>>> for seq in pyfastx.Fasta('test/data/test.fa.gz'):
>>>     print(seq.name)
>>>     print(seq.seq)
>>>     print(seq.description)
```

Iteration with `build_index=True` (default) return sequence object which allows you to access attributes of sequence. New in pyfastx 0.6.3.

3.3 Get FASTA information

```
>>> # get sequence counts in FASTA
>>> len(fa)
211

>>> # get total sequence length of FASTA
>>> fa.size
86262

>>> # get GC content of DNA sequences in FASTA
>>> fa.gc_content
43.529014587402344

>>> # get GC skew of DNA sequences in FASTA
>>> # New in pyfastx 0.3.8
>>> fa.gc_skews
0.004287730902433395

>>> # get composition of nucleotides in FASTA
>>> fa.composition
{'A': 24534, 'C': 18694, 'G': 18855, 'T': 24179}

>>> # get fasta type (DNA, RNA, or protein)
>>> # New in pyfastx 0.5.4
>>> fa.type
'DNA'

>>> # check fasta file is gzip compressed
>>> # New in pyfastx 0.5.4
>>> fa.is_gzip
True
```

3.4 Get longest and shortest sequence

New in `pyfastx` 0.3.0

```
>>> # get longest sequence
>>> s = fa.longest
>>> s
<Sequence> JZ822609.1 with length of 821

>>> s.name
'JZ822609.1'

>>> len(s)
```

(continues on next page)

(continued from previous page)

```
821

>>> # get shortest sequence
>>> s = fa.shortest
>>> s
<Sequence> JZ822617.1 with length of 118

>>> s.name
'JZ822617.1'

>>> len(s)
118
```

3.5 Calculate N50 and L50

New in pyfastx 0.3.0

Calculate assembly N50 and L50, return (N50, L50), learn more about N50,L50

```
>>> # get FASTA N50 and L50
>>> fa.nl(50)
(516, 66)

>>> # get FASTA N90 and L90
>>> fa.nl(90)
(231, 161)

>>> # get FASTA N75 and L75
>>> fa.nl(75)
(365, 117)
```

3.6 Get sequence mean and median length

New in pyfastx 0.3.0

```
>>> # get sequence average length
>>> fa.mean
408

>>> # get sequence median length
>>> fa.median
430
```

3.7 Get sequence counts

New in pyfastx 0.3.0

Get counts of sequences whose length >= specified length

```
>>> # get counts of sequences with length >= 200 bp
>>> fa.count(200)
173

>>> # get counts of sequences with length >= 500 bp
>>> fa.count(500)
70
```

3.8 Get subsequences

Subsequences can be retrieved from FASTA file by using a list of [start, end] coordinates

```
>>> # get subsequence with start and end position
>>> interval = (1, 10)
>>> fa.fetch('JZ822577.1', interval)
'CTCTAGAGAT'

>>> # get subsequences with a list of start and end position
>>> intervals = [(1, 10), (50, 60)]
>>> fa.fetch('JZ822577.1', intervals)
'CTCTAGAGATTTTAGTTTGAC'

>>> # get subsequences with reverse strand
>>> fa.fetch('JZ822577.1', (1, 10), strand='-')
'ATCTCTAGAG'
```

3.9 Get flank sequences

New in pyfastx 0.7.0

Get flank sequences for the given subsequence, return a tuple with left and right flank sequence

```
>>> # get flank sequences with length of 20 for subsequence JZ822577:50-60
>>> fa.flank('JZ822577.1', 50, 60, 20)
('TCACTCAGGCTCTTTGTCAT', 'TAGGATATCGAGTATTCAAG')

>>> # get flank sequences for a single base or SNP at position 100
>>> fa.flank('JZ822577.1', 100, 100, 20)
('GCTCATCGCTTTTGGTAATC', 'TTGCGGTGCATGCCTTTGCA')

>>> # get flank sequences by buffer cache
>>> fa.flank('JZ822577.1', 70, 90, flank_length=20, use_cache=True)
('TTTAGTTTGACTAGGATATC', 'TTGGTAATCTTTGCGGTGCA')
```

Note: The start and end position of subsequence were 1-based. When extracting flank for large numbers of subsequences from the same sequence, `use_cache=True` was recommended to improve speed.

3.10 Key function

New in `pyfastx` 0.5.1

Sometimes your fasta will have a long header which contains multiple identifiers and description, for example:

```
>JZ822577.1 contig1 cDNA library of flower petals in tree peony by suppression
subtractive hybridization Paeonia suffruticosa cDNA, mRNA sequence
```

In this case, either "JZ822577.1" or "contig1" could be used as the identifier. You can specify the key function to select one as identifier.

```
>>> #default use JZ822577.1 as identifier
>>> #specify key_func to select contig1 as identifer
>>> fa = pyfastx.Fasta('tests/data/test.fa.gz', key_func=lambda x: x.split()[1])
>>> fa
<Fasta> tests/data/test.fa.gz contains 211 seqs
```

Note: If the index file already existed, you should delete the previous index file, and then use key_func to create a new index file

CHAPTER

FOUR

SEQUENCE

4.1 Get a sequence from FASTA

```
>>> # get sequence like dictionary
>>> s1 = fa['JZ822577.1']
>>> s1
<Sequence> JZ822577.1 with length of 333

>>> # get sequence like list
>>> s2 = fa[2]
>>> s2
<Sequence> JZ822579.1 with length of 176

>>> # get last sequence
>>> s3 = fa[-1]
>>> s3
<Sequence> JZ840318.1 with length of 134

>>> # check name weather in FASTA file
>>> 'JZ822577.1' in fa
True
```

4.2 Get sequence information

```
>>> s = fa[-1]
>>> s
<Sequence> JZ840318.1 with length of 134

>>> # get sequence order number in FASTA file
>>> # New in pyfastx 0.3.7
>>> s.id
211

>>> # get sequence name
>>> s.name
'JZ840318.1'

>>> # get sequence description, New in pyfastx 0.3.1
```

(continues on next page)

(continued from previous page)

```
>>> s.description
'R283 cDNA library of flower petals in tree peony by suppression subtractive␣
↪hybridization Paeonia suffruticosa cDNA, mRNA sequence'

>>> # get sequence string
>>> s.seq

↪'ACTGGAGGTTCTTCTTCCTGTGGAAAGTAACTTGTTTTGCCTTCACCTGCCTGTTCTTCACATCAACCTTGTTCCCACACAAAACAATGGGAATGTTCTCA
↪'

>>> # get sequence raw string, New in pyfastx 0.6.3
>>> print(s.raw)
>JZ840318.1 R283 cDNA library of flower petals in tree peony by suppression subtractive␣
↪hybridization Paeonia suffruticosa cDNA, mRNA sequence
ACTGGAGGTTCTTCTTCCTGTGGAAAGTAACTTGTTTTGCCTTCACCTGCCTGTTCTTCACATCAACCTT
GTTCCCACACAAAACAATGGGAATGTTCTCACACACCCTGCAGAGATCACGATGCCATGTTGGT

>>> # get sequence length
>>> len(s)
134

>>> # get GC content if dna sequence
>>> s.gc_content
46.26865768432617

>>> # get nucleotide composition if dna sequence
>>> s.composition
{'A': 31, 'C': 37, 'G': 25, 'T': 41, 'N': 0}
```

4.3 Sequence slice

Sequence object can be sliced like a python string

```
>>> # get a sub seq from sequence
>>> s = fa[-1]
>>> ss = s[10:30]
>>> ss
<Sequence> JZ840318.1 from 11 to 30

>>> ss.name
'JZ840318.1:11-30'

>>> ss.seq
'CTTCTTCCTGTGGAAAGTAA'

>>> ss = s[-10:]
>>> ss
<Sequence> JZ840318.1 from 125 to 134

>>> ss.name
```

(continues on next page)

(continued from previous page)

```
'JZ840318.1:125-134'

>>> ss.seq
'CCATGTTGGT'
```

Note: Slicing start and end coordinates are 0-based. Currently, pyfastx does not support an optional third `step` or `stride` argument. For example `ss[::-1]`

4.4 Reverse and complement sequence

```
>>> # get sliced sequence
>>> fa[0][10:20].seq
'GTCAATTTCC'

>>> # get reverse of sliced sequence
>>> fa[0][10:20].reverse
'CCTTTAACTG'

>>> # get complement of sliced sequence
>>> fa[0][10:20].complement
'CAGTTAAAGG'

>>> # get reversed complement sequence, corresponding to sequence in antisense strand
>>> fa[0][10:20].antisense
'GGAAATTGAC'
```

4.5 Read sequence line by line

New in `pyfastx` 0.3.0

The sequence object can be iterated line by line as they appear in FASTA file.

```
>>> for line in fa[0]:
...     print(line)
...
CTCTAGAGATTACTTCTTCACATTCCAGATCACTCAGGCTCTTTGTCATTTTAGTTTGACTAGGATATCG
AGTATTCAAGCTCATCGCTTTTGGTAATCTTTGCGGTGCATGCCTTTGCATGCTGTATTGCTGCTTCATC
ATCCCCTTTGACTTGTGTGGCGGTGGCAAGACATCCGAAGAGTTAAGCGATGCTTGTCTAGTCAATTTCC
CCATGTACAGAATCATTGTTGTCAATTGGTTGTTTCCTTGATGGTGAAGGGGCTTCAATACATGAGTTCC
AAACTAACATTTCTTGACTAACACTTGAGGAAGAAGGACAAGGGTCCCCATGT
```

Note: Sliced sequence (e.g. fa[0][10:50]) cannot be read line by line

4.6 Search for subsequence

New in `pyfastx` 0.3.6

Search for subsequence from given sequence and get one-based start position of the first occurrence

```
>>> # search subsequence in sense strand
>>> fa[0].search('GCTTCAATACA')
262

>>> # check subsequence weather in sequence
>>> 'GCTTCAATACA' in fa[0]
True

>>> # search subsequence in antisense strand
>>> fa[0].search('CCTCAAGT', '-')
301
```

CHAPTER

FIVE

FASTQ

5.1 Read FASTQ file

Read plain or gzipped file and build index, support for random access to reads from FASTQ.

```
>>> import pyfastx
>>> fq = pyfastx.Fastq('tests/data/test.fq.gz')
>>> fq
<Fastq> tests/data/test.fq.gz contains 100 reads
```

5.2 FASTQ records iteration

The fastest way to parse plain or gzipped FASTQ file without building index, the iteration will return a tuple contains read name, seq and quality.

```
>>> import pyfastx
>>> for name,seq,qual in pyfastx.Fastq('tests/data/test.fq.gz', build_index=False):
>>>     print(name)
>>>     print(seq)
>>>     print(qual)
```

If you want to use full header line as read identifier without building index, you can do like this:

New in `pyfastx` 0.8.0

```
>>> import pyfastx
>>> for name,seq,qual in pyfastx.Fastq('test/data/test.fq', build_index=False, full_
↪name=True):
>>>     print(name, seq, qual)
```

You can also iterate read object from FASTQ object like this:

```
>>> import pyfastx
>>> for read in pyfastx.Fastq('test/data/test.fq.gz'):
>>>     print(read.name)
>>>     print(read.seq)
>>>     print(read.qual)
>>>     print(read.quali)
```

Iteration with `build_index=True` (default) return read object which allows you to access attribution of read. New in pyfastx 0.6.3.

5.3 Get FASTQ information

```
>>> # get read counts in FASTQ
>>> len(fq)
800

>>> # get total bases
>>> fq.size
120000

>>> # get GC content of FASTQ file
>>> fq.gc_content
66.17471313476562

>>> # get composition of bases in FASTQ
>>> fq.composition
{'A': 20501, 'C': 39705, 'G': 39704, 'T': 20089, 'N': 1}

>>> # get phred which affects the quality score conversion
>>> fq.phred
33

>>> # get the average length of reads
>>> fq.avglen
150.0

>>> # get the maximum length of reads
>>> fq.maxlen
150

>>> # get the minimum length of reads
>>> fq.minlen
150

>>> # get the maximum quality score of bases
>>> fq.maxqual
70

>>> # get the minimum quality score of bases
>>> fq.minqual
35

>>> # Guess fastq quality encoding system
>>> # New in pyfastx 0.4.1
>>> fq.encoding_type
['Sanger Phred+33', 'Illumina 1.8+ Phred+33']
```

CHAPTER

SIX

READ

6.1 Get read from FASTQ

```
>>> #get read like a dict by read name
>>> r1 = fq['A00129:183:H77K2DMXX:1:1101:4752:1047']
>>> r1
<Read> A00129:183:H77K2DMXX:1:1101:4752:1047 with length of 150

>>> # get read like a list by index
>>> r2 = fq[10]
>>> r2
<Read> A00129:183:H77K2DMXX:1:1101:18041:1078 with length of 150

>>> # get the last read
>>> r3 = fq[-1]
>>> r3
<Read> A00129:183:H77K2DMXX:1:1101:31575:4726 with length of 150

>>> # check a read weather in FASTQ file
>>> 'A00129:183:H77K2DMXX:1:1101:4752:1047' in fq
True
```

6.2 Get read information

```
>>> r = fq[-10]
>>> r
<Read> A00129:183:H77K2DMXX:1:1101:1750:4711 with length of 150

>>> # get read order number in FASTQ file
>>> r.id
791

>>> # get read name
>>> r.name
'A00129:183:H77K2DMXX:1:1101:1750:4711'

>>> # get read full header line, New in pyfastx 0.6.3
>>> r.description
```

(continues on next page)

(continued from previous page)

```
'@A00129:183:H77K2DMXX:1:1101:1750:4711 1:N:0:CAATGGAA+CGAGGCTG'

>>> # get read length
>>> len(r)
150

>>> # get read sequence
>>> r.seq

↪'CGAGGAAATCGACGTCACCGATCTGGAAGCCCTGCGCGCCCATCTCAACCAGAAATGGGGTGGCCAGCGCGGCAAGCTGACCCTGCTGCCGTTCCTGGTC
↪'

>>> # get raw string of read, New in pyfastx 0.6.3
>>> print(r.raw)
@A00129:183:H77K2DMXX:1:1101:1750:4711 1:N:0:CAATGGAA+CGAGGCTG
CGAGGAAATCGACGTCACCGATCTGGAAGCCCTGCGCGCCCATCTCAACCAGAAATGGGGTGGCCAGCGCGGCAAGCTGACCCTGCTGCCGTTCCTGGTCCGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
↪FFFFFFFFFFFFFFFFFFFFFFFFF,F:FFFFFFFFF:

>>> # get read quality ascii string
>>> r.qual

↪'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
↪FFFFFFFFFFFFFFFFFFFFFFFFF,F:FFFFFFFFF:'

>>> # get read quality integer value, ascii - 33 or 64
>>> r.quali
[37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,␣
↪37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
↪ 37, 37, 37, 37, 37, 37, 25, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,␣
↪37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
↪ 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 25,␣
↪37, 37, 11, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
↪ 37, 37, 37, 37, 37, 37, 37, 11, 37, 25, 37, 37, 37, 37, 37, 37, 37, 37, 37, 25]

>>> # get read length
>>> len(r)
150
```

CHAPTER

SEVEN

FASTAKEYS

7.1 Get fasta keys

Get all names of sequence as a list-like object

```
>>> ids = fa.keys()
>>> ids
<FastaKeys> contains 211 keys

>>> # get count of sequence
>>> len(ids)
211

>>> # get key by index
>>> ids[0]
'JZ822577.1'

>>> # check key whether in fasta
>>> 'JZ822577.1' in ids
True

>>> # iterate over keys
>>> for name in ids:
>>>     print(name)

>>> # convert to a list
>>> list(ids)
```

7.2 Sort keys

Sort keys by sequence id, name, or length for iteration

New in `pyfastx` 0.5.0

```
>>> # sort keys by length with descending order
>>> for name in ids.sort(by='length', reverse=True):
>>>     print(name)

>>> # sort keys by name with ascending order
```

(continues on next page)

(continued from previous page)

```
>>> for name in ids.sort(by='name'):
>>>     print(name)

>>> # sort keys by id with descending order
>>> for name in ids.sort(by='id', reverse=True)
>>>     print(name)
```

7.3 Filter keys

Filter keys by sequence length and name

New in pyfastx 0.5.10

```
>>> # get keys with length > 600
>>> ids.filter(ids > 600)
<FastaKeys> contains 48 keys

>>> # get keys with length >= 500 and <= 700
>>> ids.filter(ids>=500, ids<=700)
<FastaKeys> contains 48 keys

>>> # get keys with length > 500 and < 600
>>> ids.filter(500<ids<600)
<FastaKeys> contains 22 keys

>>> # get keys contain JZ8226
>>> ids.filter(ids % 'JZ8226')
<FastaKeys> contains 90 keys

>>> # get keys contain JZ8226 with length > 550
>>> ids.filter(ids % 'JZ8226', ids>550)
<FastaKeys> contains 17 keys

>>> # list a filtered result
>>> ids.filter(ids % 'JZ8226', ids>730)
>>> list(ids)
['JZ822609.1', 'JZ822650.1', 'JZ822664.1', 'JZ822699.1']

>>> # list a filtered result with sort order
>>> ids.filter(ids % 'JZ8226', ids>730).sort('length', reverse=True)
>>> list(ids)
['JZ822609.1', 'JZ822699.1', 'JZ822664.1', 'JZ822650.1']

>>> ids.filter(ids % 'JZ8226', ids>730).sort('name', reverse=True)
>>> list(ids)
['JZ822699.1', 'JZ822664.1', 'JZ822650.1', 'JZ822609.1']
```

7.4 Clear filter and sort order

```
>>> # clear sort order and filters
>>> ids.reset()
<Identifier> contains 211 identifiers
```

CHAPTER

EIGHT

FASTQKEYS

New in pyfastx 0.8.0.

8.1 Get fastq keys

Get all names of read as a list-like object.

```
>>> ids = fq.keys()
>>> ids
<FastqKeys> contains 800 keys

>>> # get count of read
>>> len(ids)
800

>>> # get key by index
>>> ids[0]
'A00129:183:H77K2DMXX:1:1101:6804:1031'

>>> # check key whether in fasta
>>> 'A00129:183:H77K2DMXX:1:1101:14416:1031' in ids
True
```

CHAPTER

NINE

COMMAND LINE INTERFACE

New in pyfastx 0.5.0

```
$ pyfastx -h

usage: pyfastx COMMAND [OPTIONS]

A command line tool for FASTA/Q file manipulation

optional arguments:
  -h, --help     show this help message and exit
  -v, --version  show program's version number and exit

Commands:

    index        build index for fasta/q file
    stat         show detailed statistics information of fasta/q file
    split        split fasta/q file into multiple files
    fq2fa        convert fastq file to fasta file
    subseq       get subsequences from fasta file by region
    sample       randomly sample sequences from fasta or fastq file
    extract      extract full sequences or reads from fasta/q file
```

9.1 Build index

New in pyfastx 0.6.10

```
$ pyfastx index -h

usage: pyfastx index [-h] [-f] fastx [fastx ...]

positional arguments:
  fastx       fasta or fastq file, gzip support

optional arguments:
  -h, --help  show this help message and exit
  -f, --full  build full index, base composition will be calculated
```

The –full option was used to count bases in FASTA/Q file and speedup calculation of GC content.

9.2 Show statistics information

```
$ pyfastx stat -h

usage: pyfastx stat [-h] fastx [fastx ...]

positional arguments:
  fastx       input fasta or fastq file, gzip support

optional arguments:
  -h, --help  show this help message and exit
```

For example:

```
$ pyfastx info tests/data/*.fa*

fileName        seqType seqCounts       totalBases         GC%   avgLen medianLen       ␣
↪maxLen  minLen  N50     L50
protein.fa      protein         17             2265           -   133.24      80.0       ␣
↪  419      23  263       4
rna.fa              RNA          2              720      65.283    360.0     360.0       ␣
↪  360     360  360       1
test.fa             DNA        211            86262      43.529   408.82     386.0       ␣
↪  821     118  516      66
test.fa.gz          DNA        211            86262      43.529   408.82     386.0       ␣
↪  821     118  516      66
```

seqType: sequence type (DNA, RNA, or protein); seqCounts: total sequence counts; totalBases: total number of bases; GC%: GC content; avgLen: average sequence length; medianLen: median sequence length; maxLen: maximum sequence length; minLen: minimum sequence length; N50: N50 length; L50: L50 sequence counts.

```
$ pyfastx info tests/data/*.fq*

fileName    readCounts  totalBases     GC%  avgLen  maxLen  minLen  maxQual minQual      ␣
↪               qualEncodingSystem
test.fq            800      120000  66.175   150.0     150     150       70      35␣
↪Sanger Phred+33,Illumina 1.8+ Phred+33
test.fq.gz         800      120000  66.175   150.0     150     150       70      35␣
↪Sanger Phred+33,Illumina 1.8+ Phred+33
```

readCounts: total read counts; totalBases: total number of bases; GC%: GC content; avgLen: average sequence length; maxLen: maximum sequence length; minLen: minimum sequence length; maxQual: maximum quality score; minQual: minimum quality score; qualEncodingSystem: quality encoding system.

9.3 Split FASTA/Q file

```
$ pyfastx split -h

usage: pyfastx split [-h] (-n int | -c int) [-o str] fastx

positional arguments:
  fastx                 fasta or fastq file, gzip support

optional arguments:
  -h, --help            show this help message and exit
  -n int                split a fasta/q file into N new files with even size
  -c int                split a fasta/q file into multiple files containing the same␣
↪sequence counts
  -o str, --out-dir str
                        output directory, default is current folder
```

9.4 Convert FASTQ to FASTA file

```
$ pyfastx fq2fa -h

usage: pyfastx fq2fa [-h] [-o str] fastx

positional arguments:
  fastx                 input fastq file, gzip support

optional arguments:
  -h, --help            show this help message and exit
  -o str, --out-file str
                        output file, default: output to stdout
```

9.5 Get subsequence with region

```
$ pyfastx subseq -h

usage: pyfastx subseq [-h] [-r str | -b str] [-o str]
                      fastx [region [region ...]]

positional arguments:
  fastx                 input fasta file, gzip support
  region                format is chr:start-end, start and end position is
                        1-based, multiple regions were separated by space

optional arguments:
  -h, --help            show this help message and exit
  -r str, --region-file str
                        tab-delimited file, one region per line, both start
                        and end position are 1-based
```

(continues on next page)

(continued from previous page)

```
  -b str, --bed-file str
                        tab-delimited BED file, 0-based start position and
                        1-based end position
  -o str, --out-file str
                        output file, default: output to stdout
```

9.6 Sample sequences

```
$ pyfastx sample -h

usage: pyfastx sample [-h] (-n int | -p float) [-s int] [--sequential-read]
                      [-o str]
                      fastx

positional arguments:
  fastx                 fasta or fastq file, gzip support

optional arguments:
  -h, --help            show this help message and exit
  -n int                number of sequences to be sampled
  -p float              proportion of sequences to be sampled, 0~1
  -s int, --seed int    random seed, default is the current system time
  --sequential-read     start sequential reading, particularly suitable for
                        sampling large numbers of sequences
  -o str, --out-file str
                        output file, default: output to stdout
```

9.7 Extract sequences

New in pyfastx 0.6.10

```
$ pyfastx extract -h

usage: pyfastx extract [-h] [-l str] [--reverse-complement] [--out-fasta]
                       [-o str] [--sequential-read]
                       fastx [name [name ...]]

positional arguments:
  fastx                 fasta or fastq file, gzip support
  name                  sequence name or read name, multiple names were
                        separated by space

optional arguments:
  -h, --help            show this help message and exit
  -l str, --list-file str
                        a file containing sequence or read names, one name per
                        line
  --reverse-complement  output reverse complement sequence
```

(continues on next page)

(continued from previous page)

```
  --out-fasta           output fasta format when extract reads from fastq,
                        default output fastq format
  -o str, --out-file str
                        output file, default: output to stdout
  --sequential-read     start sequential reading, particularly suitable for
                        extracting large numbers of sequences
```

CHAPTER

TEN

MULTIPLE PROCESSES

Pyfastx can be used with multiprocessing module to speed up the random access. Prior to reading sequences from subprocesses, you have to ensure that index file has been created from main process. The index file of pyfastx is just a SQLite3 database file which supports for concurrency. We provide two simple examples for using pyfastx with multiprocessing pool.

10.1 Example one

```
import random
import pyfastx
import multiprocessing as mp

# process worker
# randomly fetch five sequences and print to stdout
def worker(woker_num, seq_counts):
    #recreate the Fasta object in subprocess
    fa = pyfastx.Fasta('test.fa')

    for i in random.sample(range(seq_counts), 5):
        print("worker {} print:\n{}".format(worker_num, fa[i].raw))

if __name__ == '__main__':
    #ensure index file has been created in main process
    fa = pyfastx.Fasta('test.fa')

    #get sequence counts
    c = len(fa)

    #start the process pool
    pool = mp.Pool()

    #add five task workers to run
    for n in range(5):
        pool.apply_async(worker, args=(n, c))

    #wait for tasks to finish
    pool.close()
    pool.join()
```

CHAPTER

ELEVEN

DRAWBACKS

If you intensively check sequence names exists in FASTA file using `in` operator on FASTA object like:

```
>>> fa = pyfastx.Fasta('tests/data/test.fa.gz')
>>> # Suppose seqnames has 100000 names
>>> for seqname in seqnames:
>>>     if seqname in fa:
>>>             do something
```

This will take a long time to finish. Because, pyfastx does not load the index into memory, the `in` operating is corresponding to sql query existence from index database. The faster alternative way to do this is:

```
>>> fa = pyfastx.Fasta('tests/data/test.fa.gz')
>>> # load all sequence names into a set object
>>> all_names = set(fa.keys())
>>> for seqname in seqnames:
>>>     if seqname in all_names:
>>>             do something
```

CHAPTER

TWELVE

CHANGELOG

12.1 Version 0.8.4 (2021-06-30)

- Added slice feature to FastaKeys
- Fixed FastaKeys and FastqKeys iteration memory leak
- Optimized FastaKeys and FastqKeys creation

12.2 Version 0.8.3 (2021-04-25)

- Fixed Fastx iteration for next function
- Fixed Fastx uppercase for reading fasta

12.3 Version 0.8.2 (2021-01-02)

- Fixed sample segfault error caused by fastq iteration error
- Fixed gzip index import error in multiple processes
- Fixed fastq iteration segfault error with full_name=True
- Fixed all objects iteration to support built-in next function

12.4 Version 0.8.1 (2020-12-16)

- Fixed pip install error from source code
- Removed support for python39 32bit due to dll load error

12.5 Version 0.8.0 (2020-12-15)

- Added Fastx object as a simple sequence iterator
- Added FastqKeys object to obtain read names
- Added full_name option to Fastq object
- Added support for Python 3.9
- Fixed Fasta object error identifier order
- Optimized speed of containing test and iteration
- Changed Identifier object to FastaKeys object

12.6 Version 0.7.0 (2020-09-20)

- Added support for extracting flank sequences
- Added support for indexing super large gzip file
- Reduced memory consumption when building gzip index
- Improved the speed of random access to reads from fastq
- Fixed sequence dealloc error cuasing no fasta delloc trigger
- Fixed fastq max and min quality score return value

12.7 Older versions

12.7.1 Version 0.6.17 (2020-08-31)

- Fixed gzip index loading error when no write permission

12.7.2 Version 0.6.16 (2020-08-27)

- Increased the buff size of kseq to speedup sequence iteration
- Removed warning message from fasta.c when building full index

12.7.3 Version 0.6.15 (2020-08-25)

- Fixed key_func error caused by free operation
- Fixed full name error when reading sequence without whitespace in names
- Fixed a hidden bug in fasta/q iteration when reading attributes (not seq)
- Fixed fasta/fastq size and sequence count error on Windows when parsing large file
- Fixed zlib 2gb and 4gb limit on windows x64 to support large file
- Reduced seek point span size to speedup random access from gzip file

12.7.4 Version 0.6.14 (2020-07-31)

- Added support for using full header as identifier without building index
- Improved the speed of fasta sequence iteration
- Improved the speed of gzipped fastq read iteration
- Fixed a bug in fastq read reader

12.7.5 Version 0.6.13 (2020-07-09)

- Fixed fastq read iteration error
- Fixed fastq cache buffer reader
- Added cache for mean, median and N50 length
- Speedup fasta iteration by reduced seeks

12.7.6 Version 0.6.12 (2020-06-14)

- Fixed DeprecationWarning on py38 caused by '#' formats args
- Fixed some memory leak bugs
- Cached sequence name to speedup fetch method
- Used random string as gzip index temp file to support multiple processes

12.7.7 Version 0.6.11 (2020-05-18)

- Fixed iteration error on Windows
- Fixed test error on Windows
- Fixed fastq composition error on 32bit OS
- Improved the speed of fasta identifier sort and filter

12.7.8 Version 0.6.10 (2020-04-22)

- Improved the speed of sequence reading
- Improved the speed of sequence line iteration
- Added avglen, minlen, maxlen, minqual and maxqual to Fastq object
- Fixed read retrieval error
- Fixed some hidden memory leaks
- Changed fastq index file structure to save more information

12.7.9 Version 0.6.9 (2020-04-12)

- Added buffreader to improve speed for reading from gzipped file
- Added extract subcommand to extract sequences from fasta/q file
- Added build subcommand to just build index
- Changed info subcommand output to a tab seperated table
- Changed Fastq object composition parameter to full_index

12.7.10 Version 0.6.8 (2020-03-14)

- Fixed large offset seek error on windows
- Fixed PyUnicode_AsUTF8 const char type warning
- Changed sequence read line by line function
- Changed gzread to fread for fastq information

12.7.11 Version 0.6.7 (2020-03-03)

- Added check for fasta/q format when open file
- Added benchmark scripts for evaluating performance
- Speed up the fasta/q object iteration
- Optimzed str length warning caused by strlen

12.7.12 Version 0.6.6 (2020-02-15)

- Fixed incorrect sliced sequence name
- Fixed seq,identifier,read object memory dealloc
- Changed description text into description length in index file

12.7.13 Version 0.6.5 (2020-01-31)

- Reduced memory usage when building index for large fasta
- Removed rebuild_index method from Fasta object due to segmentation fault
- Optimized compatibility between sqlite3 and python GIL

12.7.14 Version 0.6.4 (2020-01-14)

- Fixed last sequence fetching error caused by missing n
- Improved fasta/q object key error message to make it more human

12.7.15 Version 0.6.3 (2020-01-08)

- Added .raw attribute to sequence object to get seq raw string
- Added .raw attribute to read object to get read raw string
- Added .description to read object to get full header line
- Added iteration for sequence object from FASTA object
- Added iteration for tuple from FASTQ object
- Changed FASTA class parameter composition to full_index

12.7.16 Version 0.6.2 (2020-01-04)

- Fixed sample sequence index error
- Fixed ci deploy error

12.7.17 Version 0.6.1 (2020-01-03)

- Added sample sequences command line
- Added get subsequence command line

12.7.18 Version 0.6.0 (2020-01-02)

- Fixed FASTA object parameter error
- Fixed identifier sprintf warning
- Fixed fasta description end r retained
- Fixed error byte length when slice sequence
- Removed support for python2.7 and python3.4
- Removed python2 compat
- Disabled export gzip index when building memory index

12.7.19 Version 0.5.10 (2019-11-20)

- Added identifier filter function
- Remove tp_new for Read, Sequence and Identifier
- Fixed module method error

12.7.20 Version 0.5.9 (2019-11-17)

- Added get longest and shortest sequence object
- Added composition argument to speedup getting GC content
- Added memory index to keep index in memory rather than local file
- Fixed command line error
- Changed sqlite to higher version
- Removed journal_mode OFF
- Speedup index building

12.7.21 Version 0.5.8 (2019-11-10)

- Fixed fasta NL function parameter check
- Fixed read id error when fastq iteration

12.7.22 Version 0.5.7 (2019-11-09)

- Fixed SystemError caused caused by Python 2.7 seperated int and long type
- Fixed String type check on Python 2.7
- Fixed objects memory deallocation

12.7.23 Version 0.5.6 (2019-11-08)

- Optimized random access from plain file
- Reduced memory consumption

12.7.24 Version 0.5.5 (2019-11-07)

- Added Support for IUPAC code complement
- Speedup reverse complement
- Speedup space removing and uppercase

12.7.25 Version 0.5.4 (2019-11-04)

- Added guess fasta type (DNA, RNA, protein)
- Added support for calculating protein sequence composition
- Optimized the speed of index building
- Calculate sequence composition when get gc content or composition
- Fixed char return in python 2.7

12.7.26 Version 0.5.3 (2019-10-23)

- Added support for coverting fastq to fasta
- Updated command line interface docs
- Fixed command line entry points

12.7.27 Version 0.5.2 (2019-10-18)

- Fixed command line interface running error

12.7.28 Version 0.5.1 (2019-10-17)

- Added key function for custom sequence identifier
- Optimized speed of fasta indexing
- Fixed bool args parsing error in py2.7

12.7.29 Version 0.5.0 (2019-10-13)

- Added support for python 2.7 and 3.4
- Added command line tool to manipulate fasta and fastq file
- Added gzip attribute to fasta and fastq object to check whether compressed
- Added sort function for identifier object
- Fixed python bool argument parsing error caused by uint16_t
- Fixed identifier sort key initialization

12.7.30 Version 0.4.1 (2019-10-05)

- Fixed fastq quality encoding system guesser
- Fixed gzip index insertion error

12.7.31 Version 0.4.0 (2019-09-29)

- Added support for parsing FASTQ
- Added random access to reads from FASTQ

12.7.32 Version 0.3.10 (2019-09-27)

- Fixed GC skew exception caused by mixing unsigned with signed for division

12.7.33 Version 0.3.9 (2019-09-26)

- Fixed sequence read line by line error
- Fixed last sequence build index error when fasta file ended without n
- Fixed GC skew error

12.7.34 Version 0.3.8 (2019-09-25)

- Fixed large offset became negative error
- Fixed slice step
- Fixed uncorrect median length
- Fixed strand compare error
- Added GC skew calculation
- Updated test script

12.7.35 Version 0.3.7 (2019-09-24)

- Changed int type to standard type
- Added support for processing large fasta file
- Added id number for each sequence
- Fixed SQL fetch error
- Used 50 as default value of nl to calculate N50 and L50

12.7.36 Version 0.3.6 (2019-09-20)

- Added support for searching subsequence from a sequence
- Added support for checking subsequence weather in a sequence
- Fixed gzip index import error
- Fixed subsequence parent length for full sequence extraction

12.7.37 Version 0.3.5 (2019-09-08)

- Fixed unicode error caused by sqlite3_finalize

12.7.38 Version 0.3.4 (2019-09-07)

- Fixed seq description unicode string error

12.7.39 Version 0.3.3 (2019-09-07)

- Fixed sequence description encoding error

12.7.40 Version 0.3.2 (2019-09-07)

Deleted

12.7.41 Version 0.3.1 (2019-09-07)

- Added support for geting sequence description

12.7.42 Version 0.3.0 (2019-09-07)

- Added read sequence from fasta file line by line
- Added support for calculating assembly N50 and L50
- Added support for calculating median and average length
- Added support for getting longest and shortest sequence
- Added support for calculating counts of sequence
- removed support for Python34

12.7.43 Version 0.2.11 (2019-08-31)

- Support for Python 3.4

12.7.44 Version 0.2.10 (2019-08-28)

- Changed fseek and fread into gzseek and gzread
- Fixed sequence cache name comparision
- Fixed last sequence read error without line end
- Fixed subsequence slice error in normal FASTA file

12.7.45 Version 0.2.9 (2019-08-27)

- Fixed bad line calculate error
- Changed rewind to fseek for subsequence extraction

12.7.46 Version 0.2.8 (2019-08-26)

- Changed kseq.h library from li to attractivechaos
- Improved fasta parser

12.7.47 Version 0.2.7 (2019-08-26)

- Fixed no gzip index wrote to sqlite index file

12.7.48 Version 0.2.6 (2019-08-26)

- Optimized speed of gzip random access

12.7.49 Version 0.2.5 (2019-08-25)

- Fixed segmentation fault raised when loading gzip index
- Changed fasta object method get_seq to fetch

12.7.50 Version 0.2.4 (2019-08-25)

- Fixed fasta iter error after building new index

12.7.51 Version 0.2.3 (2019-08-24)

- Fixed fasta iter error when end of file is not n

12.7.52 Version 0.2.2 (2019-07-19)

- Fixed identifier contain error

12.7.53 Version 0.2.1 (2019-07-15)

- Fixed sequence name always end with 0
- Fixed fasta iterable for flat fasta

12.7.54 Version 0.2.0 (2019-07-09)

- First release to PyPI

CHAPTER

THIRTEEN

API REFERENCE

13.1 pyfastx.version

pyfastx.**version**(*debug=False*)

Get current version of pyfastx

Parameters
debug (*bool*) – if true, return versions of pyfastx, zlib, sqlite3 and zran.

Returns
version of pyfastx

Return type
str

pyfastx.**gzip_check**(*file_name*)

New in pyfastx 0.5.4

Check file is gzip compressed or not

Parameters
file_name (*str*) – the path of input file

Returns
Ture if file is gzip compressed else False

Return type
bool

13.2 pyfastx.Fasta

class pyfastx.**Fasta**(*file_name*, *uppercase=True*, *build_index=True*, *full_index=False*, *full_name=False*, *memory_index=False*, *key_func=None*)

Read and parse fasta files. Fasta can be used as dict or list, you can use index or sequence name to get a sequence object, e.g. `fasta[0]`, `fasta['seq1']`

Parameters

- **file_name** (*str*) – the file path of input FASTA file
- **uppercase** (*bool*) – always output uppercase sequence, default: `True`
- **build_index** (*bool*) – build index for random access to FASTA sequence, default: `True`. If build_index is False, iteration will return a tuple (name, seq); If build_index is True, iteration will return a sequence object.

- **full_index** (*bool*) – calculate character (e.g. A, T, G, C) composition when building index, this will improve the speed of GC content extracting. However, it will take more time to build index, default: `False`
- **full_name** (*bool*) – use the full header line instead of the part before first whitespace as the identifier of sequence, even in mode without building index. New in 0.6.14, default: `False`
- **memory_index** (*bool*) – if memory_index is True, the fasta index will be kept in memory and do not generate a index file, default: `False`
- **key_func** (*function*) – new in 0.5.1, key function is generally a lambda expression to split header and obtain a shortened identifer, default: `None`

Returns
Fasta object

file_name

FASTA file path

size

total length of sequences in FASTA file

type

New in `pyfastx` 0.5.4

get fasta type, return DNA, RNA, protein, or unknown

is_gzip

New in pyfastx 0.5.0

return True if fasta is gzip compressed else return False

gc_content

GC content of whole sequences in FASTA file, return a float value

gc_skew

GC skew of whole sequences in FASTA file, learn more about GC skew

New in `pyfastx` 0.3.8

composition

nucleotide composition in FASTA file, a dict contains counts of A, T, G, C and N (unkown nucleotide base)

longest

get longest sequence in FASTA file, return a Sequence object

New in `pyfastx` 0.3.0

shortest

get shortest sequence in FASTA file, return a Sequence object

New in `pyfastx` 0.3.0

mean

get average length of sequences in FASTA file

New in `pyfastx` 0.3.0

median

get median length of sequences in FASTA file

New in `pyfastx` 0.3.0

fetch(*chrom*, *intervals*, *strand='+'*)

truncate subsequences from a given sequence by a start and end coordinate or a list of coordinates. This function will cache the full sequence into memory, and is suitable for extracting large numbers of subsequences from specified sequence.

Parameters

- **chrom** (*str*) – chromosome name or sequence name
- **intervals** (*list/tuple*) – list of [start, end] coordinates
- **strand** (*str*) – sequence strand, + indicates sense strand, - indicates antisense strand, default: '+'

Note: intervals can be a list or tuple with start and end position e.g. (10, 20). intervals also can be a list or tuple with multiple coordinates e.g. [(10, 20), (50,70)]

Returns
sliced subsequences

Return type
str

flank(*chrom*, *start*, *end*, *flank_length=50*, *use_cache=False*)

Get the flank sequence of given subsequence with start and end. New in 0.7.0

Parameters

- **chrom** (*str*) – chromosome name or sequence name
- **start** (*int*) – 1-based start position of subsequence on chrom
- **end** (*int*) – 1-based end position of subsequence on chrom
- **flank_length** (*int*) – length of flank sequence, default 50
- **use_cache** (*bool*) – cache the whole sequence

Note: If you want to extract flank sequence for large numbers of subsequences from the same sequence. Use `use_cache=True` will greatly improve the speed

---

Returns
left flank and right flank sequence

Return type
tuple

build_index()

build index for FASTA file

keys()

get all names of sequences

Returns
an FastaKeys object

count(*n*)

get counts of sequences whose length >= n bp

New in pyfastx 0.3.0

Parameters
n (*int*) – number of bases

Returns
sequence counts

Return type
int

nl(*quantile*)

calculate assembly N50 and L50, learn more about N50,L50

New in pyfastx 0.3.0

Parameters
quantile (*int*) – a number between 0 and 100, default 50

Returns
(N50, L50)

Return type
tuple

13.3 pyfastx.Sequence

class pyfastx.**Sequence**

Readonly sequence object generated by fasta object, Sequence can be treated as a list and support slicing e.g. `seq[10:20]`

id

sequence id or order number in FASTA file

name

sequence name

description

Get sequence description after name in sequence header

New in pyfastx 0.3.1

start

start position of sequence

end

end position of sequence

gc_content

GC content of current sequence, return a float value

gc_skew

GC skew of current sequence, learn more about GC skew

composition
nucleotide composition of sequence, a dict contains counts of A, T, G, C and N (unkown nucleotide base)

raw
get the raw string (with header line and sequence lines) of sequence as it appeared in file

New in pyfastx 0.6.3

seq
get the string of sequence in sense strand

reverse
get the string of reversed sequence

complement
get the string of complement sequence

antisense
get the string of sequence in antisense strand, corresponding to reversed and complement sequence

search(*subseq*, *strand='+'*)
Search for subsequence from given sequence and get the start position of the first occurrence

New in pyfastx 0.3.6

Parameters

- **subseq** (*str*) – a subsequence for search
- **strand** (*str*) – sequence strand + or -, default +

Returns
if found subsequence return one-based start position, if not return None

Return type
int or None

13.4 pyfastx.Fastq

New in pyfastx 0.4.0

class pyfastx.**Fastq**(*file_name*, *phred=0*, *build_index=True*, *full_index=False*)
Read and parse fastq file

Parameters

- **file_name** (*str*) – input fastq file path
- **build_index** (*bool*) – build index for random access to FASTQ reads, default: True. If build_index is False, iteration will return a tuple (name, seq, qual); If build_index is True, iteration will return a read object
- **full_index** (*bool*) – calculate character (e.g. A, T, G, C) composition when building index, this will improve the speed of GC content extracting. However, it will take more time to build index, default: False
- **phred** (*int*) – phred was used to convert quality ascii to quality int value, usually is 33 or 64, default 33

Returns
Fastq object

file_name

FASTQ file path

size

total bases in FASTQ file

is_gzip

New in pyfastx 0.5.0

return True if fasta is gzip compressed else return False

gc_content

GC content of whole FASTQ file

avglen

New in `pyfastx` 0.6.10

get average length of reads

maxlen

New in `pyfastx` 0.6.10

get maximum length of reads

minlen

New in `pyfastx` 0.6.10

get minimum length of reads

maxqual

New in `pyfastx` 0.6.10

get maximum quality score of bases

minqual

New in `pyfastx` 0.6.10

get minimum quality score of bases

composition

base composition in FASTQ file, a dict contains counts of A, T, G, C and N (unkown nucleotide base)

phred

get phred value

encoding_type

New in `pyfastx` 0.4.1

Guess the quality encoding type used by FASTQ sequence file

build_index()

Build index for fastq file when build_index set to False

keys()

New in `pyfastx` 0.8.0

Get all the names of reads in fastq file

Returns
an FastqKeys object

13.5 pyfastx.Read

New in pyfastx 0.4.0

class pyfastx.**Read**

Readonly read object for obtaining read information, generated by fastq object

id

read id or order number in FASTQ file

name

read name excluding '@'

description

get the full header line of read

raw

get the raw string (with header, sequence, comment and quality lines) of read as it appeared in file

New in pyfastx 0.6.3

seq

get read sequence string

qual

get read quality ascii string

13.6 pyfastx.Fastx

class pyfastx.**Fastx**(*file_name*, *format='auto'*, *uppercase=False*)

New in pyfastx 0.8.0. A python binding of kseq.h, provide a simple api for iterating over sequences in fasta/q file

Parameters

- **file_name** (*str*) – input fasta or fastq file path
- **format** (*str*) – the input file format, can be "fasta" or "fastq", default: "auto", automatically detect the format of sequence file
- **uppercase** (*bool*) – always output uppercase sequence, only work for fasta file, default: False

Returns

Fastx object

13.7 pyfastx.FastaKeys

class pyfastx.**FastaKeys**

FastaKeys is a readonly and list-like object, contains all names of sequences

sort(*by='id'*, *reverse=False*)

Sort keys by sequence id, name or length for iteration

New in pyfastx 0.5.0

Parameters

- **by** (*str*) – order by id, name, or length, default is id
- **reverse** (*bool*) – used to flag descending sorts, default is False

Returns
FastaKeys object itself

filter(**filters*)

Filter keys by sequence name and length for iteration

Parameters
filters (*list*) – filters generated by comparison like ids > 500 or ids % 'seq1', where ids is a Identifier object

Returns
FastaKeys object itself

reset()

Clear all filters and sort order

Returns
FastaKeys object itself

13.8 pyfastx.FastqKeys

class pyfastx.**FastqKeys**

New in pyfastx 0.8.0. FastqKeys is a readonly and list-like object, contains all names of reads

CHAPTER

FOURTEEN

ACKNOWLEDGEMENTS

kseq.h and zlib was used to parse FASTA format. Sqlite3 was used to store built indexes. `pyfastx` can randomly access to sequences from gzipped FASTA file mainly attributed to indexed_gzip.

CHAPTER

FIFTEEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

A

antisense (*pyfastx.Sequence attribute*), 53
avglen (*pyfastx.Fastq attribute*), 54

B

build_index() (*pyfastx.Fasta method*), 51
build_index() (*pyfastx.Fastq method*), 54
built-in function
    pyfastx.gzip_check(), 49
    pyfastx.version(), 49

C

complement (*pyfastx.Sequence attribute*), 53
composition (*pyfastx.Fasta attribute*), 50
composition (*pyfastx.Fastq attribute*), 54
composition (*pyfastx.Sequence attribute*), 52
count() (*pyfastx.Fasta method*), 51

D

description (*pyfastx.Read attribute*), 55
description (*pyfastx.Sequence attribute*), 52

E

encoding_type (*pyfastx.Fastq attribute*), 54
end (*pyfastx.Sequence attribute*), 52

F

fetch() (*pyfastx.Fasta method*), 50
file_name (*pyfastx.Fasta attribute*), 50
file_name (*pyfastx.Fastq attribute*), 54
filter() (*pyfastx.FastaKeys method*), 56
flank() (*pyfastx.Fasta method*), 51

G

gc_content (*pyfastx.Fasta attribute*), 50
gc_content (*pyfastx.Fastq attribute*), 54
gc_content (*pyfastx.Sequence attribute*), 52
gc_skew (*pyfastx.Fasta attribute*), 50
gc_skew (*pyfastx.Sequence attribute*), 52

I

id (*pyfastx.Read attribute*), 55
id (*pyfastx.Sequence attribute*), 52
is_gzip (*pyfastx.Fasta attribute*), 50
is_gzip (*pyfastx.Fastq attribute*), 54

K

keys() (*pyfastx.Fasta method*), 51
keys() (*pyfastx.Fastq method*), 54

L

longest (*pyfastx.Fasta attribute*), 50

M

maxlen (*pyfastx.Fastq attribute*), 54
maxqual (*pyfastx.Fastq attribute*), 54
mean (*pyfastx.Fasta attribute*), 50
median (*pyfastx.Fasta attribute*), 50
minlen (*pyfastx.Fastq attribute*), 54
minqual (*pyfastx.Fastq attribute*), 54

N

name (*pyfastx.Read attribute*), 55
name (*pyfastx.Sequence attribute*), 52
nl() (*pyfastx.Fasta method*), 52

P

phred (*pyfastx.Fastq attribute*), 54
pyfastx.Fasta (*built-in class*), 49
pyfastx.FastaKeys (*built-in class*), 56
pyfastx.Fastq (*built-in class*), 53
pyfastx.FastqKeys (*built-in class*), 56
pyfastx.Fastx (*built-in class*), 55
pyfastx.gzip_check()
    built-in function, 49
pyfastx.Read (*built-in class*), 55
pyfastx.Sequence (*built-in class*), 52
pyfastx.version()
    built-in function, 49

Q

qual (*pyfastx.Read attribute*), 55

R

raw (*pyfastx.Read attribute*), 55
raw (*pyfastx.Sequence attribute*), 53
reset() (*pyfastx.FastaKeys method*), 56
reverse (*pyfastx.Sequence attribute*), 53

S

search() (*pyfastx.Sequence method*), 53
seq (*pyfastx.Read attribute*), 55
seq (*pyfastx.Sequence attribute*), 53
shortest (*pyfastx.Fasta attribute*), 50
size (*pyfastx.Fasta attribute*), 50
size (*pyfastx.Fastq attribute*), 54
sort() (*pyfastx.FastaKeys method*), 56
start (*pyfastx.Sequence attribute*), 52

T

type (*pyfastx.Fasta attribute*), 50