SeqKit - a cross-platform and ultrafast toolkit for FASTA/Q file manipulation

Introduction

FASTA and FASTQ are basic formats for storing nucleotide and protein sequences. The manipulations of FASTA/Q file include converting, clipping, searching, filtering, deduplication, splitting, shuffling, sampling and so on. Existed tools only implemented parts of the functions, and some of them are only available for specific operating systems. Furthermore, the complicated installation process of dependencies packages and running environment also make them less friendly to common users.

SeqKit is a cross-platform, ultrafast, and practical FASTA/Q manipulations tool that is friendly for researchers to complete wide ranges of FASTA/Q file processing. The toolkit supports plain or gzip-compressed input and output from either standard stream or files, therefore, it could be easily used in command-line pipe.

Wei Shen
Third Military Medical University, China
http://shenwei.me
Aug 2016
About

Wei Shen
PhD student of Microbiology
Third Military University, China

Biography

Wei Shen is a PhD student of Microbiology studying Bioinformatics algorithms and gut metagenomics at Third Military University, China. He also tried to apply NGS technique to improve some molecular biology problems. More: Google Scholar, ResearchGate, ORCID.

He developed and shared lots of reusable open-source tools of Bioinformatics and data manipulations at Github (also hosted at http://bioinf.shenwei.me). And he has written tens of posts about software development and scientific research. He also likes to collect awesome resources on Bioinformatics, data science, machine learning and programming language, and share talks about Linux using and software development.

http://shenwei.me
FASTA/Q formats

FASTA

>cel-let-7 MI0000001 Caenorhabditis elegans let-7 stem-loop
UACACUGUGGAUCCGGUGAGGUAGGUAGGUUGUAUAGUUUGGAUAUAUUACCACCCGUGAGAAC
UAUGCAAAUUUUUCUACCUCUACCCGGAGACAGAAGACUCUUCGA

FASTQ

@HWI-D00523:240:HF3WGBCX:1:1101:2574:2226 1:N:0:CTGTAG
TGAGGAATATTGGTCATGGGCAGCCTGAACCACCCAGCGTACGTGAGGATGACTG
+
HIHIHIHIHIIHGHHIIHIIIIIIIIIHIIIIIIHIIHIIIIIGIHHIHHHHH

Basic and ubiquitous formats for storing nucleotide and protein sequences
Common Manipulations

- Converting
- Searching
- Filtering
- Deduplication
- Splitting
- Shuffling
- Sampling
- ...

Existed Tools

- fasta_utilities
- fastx_toolkit
- pyfaidx
- seqmagick
- seqtk
- ...

Why another wheel?

Limitations

- Only implement parts of functions
- Not efficient
- Require dependencies/running environments
- Hard to install
- Only available for some operating systems
- Not user friendly
Functions already implemented in bio_scripts

- https://github.com/shenwei356/bio_scripts/tree/master/sequence
- fasta2tab/tab2fasta, fastq2tab/tab2fastq
- fasta_extract_by_pattern.pl
- fasta_common_seqs.pl
- fasta_remove_duplicates.pl
- fasta_locate_motif.pl
- gff2fa.py

Limitations

- Need running environment
- Need dependencies (Perl modules), hard for beginner
- Hard to maintain, too many files
- Scripts never live long
- I want to write something in Golang
Method
Programming Language

- **Requirements:**
  - Easy to install and use ☆☆☆
  - High performance ☆☆☆
  - Cross platform ☆☆
  - Easy for development ☆☆

- **Candidates**
  - C/C++, not good at
  - Perl/Python, fast and easy to write, but need running environment
  - Java, cross platform, fast but verbose
  - R, not suitable
  - Go, meets all
Program organization

SeqKit, a command-line toolkit

```
[shenwei@Riolinux tests]$ seqkit head -n 1 hairpin.fa
>cel-let-7 MI0000001 Caenorhabditis elegans let-7 stem-loop
UACACUGUGGAUCCGGUGAGGUGUAGUUUGUAGUUUGGAAAUUACCACCGGGUGAAC
UAUGCAAUUUUCUACCUUACCAGAGACAGAUCUCUUGA
```
FASTA/Q format parsing

- **Performance bottleneck**: file I/O
- Asynchronously parsing with buffered chunk of records
- Serially reading (reading record one by one)
- Optimizing never ends!
  - 100+ commits and 30+ releases in 5 months
  - See the long long changelogs
Performance optimization ☆☆☆☆☆

- Parallelizing CPU intensive processes. e.g. GTF/BED file parsing
- Customized data structures and algorithms, e.g. complementary nucleic base
- Golang:
  - Avoid GC! Reuse objects!
  - *Discarding standard library bufio, manage buffer by hand!*
  - Avoid using stdlib regexp!

*Optimization never ends!*
Reproducibility

- Platform-compatibility
  Results can be repeated with same datasets and parameters
- Configurability of random seed for “sample” and “shuffle”
Result
Source code

https://github.com/shenwei356/seqkit

A cross-platform and ultrafast toolkit for FASTA/Q file manipulation in Golang

http://shenwei356.github.io/seqkit — Edit

- benchmark
- doc
- seqkit
- tests
- .gitignore
- LICENSE
- README.md

Latest commit 1680862 a day ago

- update benchmark/README.md 5 days ago
- sliding output supports fastq now a day ago
- sliding output supports fastq now a day ago
- v0.3.0 17 days ago
- rename faskit to seqkit 23 days ago
- finishing subcommands: seq, extract 5 months ago
- update doc 5 days ago
Coding

> 10,000 lines source code
SeqKit - a cross-platform and ultrafast toolkit for FASTA/Q file manipulation

Introduction

FASTA and FASTQ are basic formats for storing nucleotide and protein sequences. The manipulations of FASTA/Q file include converting, clipping, searching, filtering, deduplication, splitting, shuffling, sampling and so on. Existed tools only implemented parts of the functions, and some of them are only available for specific operating systems. Furthermore, the complicated installation process of dependencies packages and running environment also make them less friendly to common users.

SeqKit is a cross-platform, ultrafast, and practical FASTA/Q manipulations tool that is friendly for researchers to complete wide ranges of FASTA/Q file processing. The toolkit supports plain or gzip-compressed input and output from either standard stream or files, therefore, it could be easily used in command-line pipe.
## Functions

### Sequence and subsequence
- **seq**  transform sequences (reverse, complement, extract ID...)
- **subseq**  get subsequences by region/gtf/bed, including flanking sequences
- **sliding**  sliding sequences, circular genome supported
- **stat**  simple statistics of FASTA files
- **faidx**  create FASTA index file

### Format conversion
- **fx2tab**  convert FASTA/Q to tabular format (and length/GC content/GC skew)
- **tab2fx**  convert tabular format to FASTA/Q format
- **fq2fa**  covert FASTQ to FASTA

### Searching
- **grep**  search sequences by pattern(s) of name or sequence motifs
- **locate**  locate subsequences/motifs

### Set operations
- **rmdup**  remove duplicated sequences by id/name/sequence
- **common**  find common sequences of multiple files by id/name/sequence
- **split**  split sequences into files by id/seq region/size/parts
- **sample**  sample sequences by number or proportion
- **head**  print first N FASTA/Q records

### Edit
- **replace**  replace name/sequence by regular expression
- **rename**  rename duplicated IDs

### Ordering
- **shuffle**  shuffle sequences
- **sort**  sort sequences by id/name/sequence

### Misc
- **version**  print version information
Features

- **Cross-platform** (Linux/Windows/Mac OS X/OpenBSD/FreeBSD)
- **Light weight and out-of-the-box, no dependencies, no compilation, no configuration**
- **Ultrafast** (see benchmark), multiple-CPU supported.
- **Practical functions** supported by 20 subcommands
- **Well documented** (detailed usage and benchmark)
- Seamlessly parses both FASTA and FASTQ formats
- **Support STDIN** and gziped input/output file, easy being used in pipe
- Support custom sequence ID regular expression (especially useful for querying with ID list)
<table>
<thead>
<tr>
<th>Categories</th>
<th>Features</th>
<th>seqkit</th>
<th>fasta_utilities</th>
<th>fastx_toolkit</th>
<th>pyfaidx</th>
<th>seqmagick</th>
<th>seqtk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Formats supports</strong></td>
<td>Multi-line FASTA</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>FASTQ</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Multi-line FASTQ</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Validating sequences</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Supporting RNA</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
<td>Searching by motifs</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Sampling</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Extracting sub-sequence</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Removing duplicates</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>Partly</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Splitting</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Partly</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Splitting by seq</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Shuffling</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Sorting</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
<td>--</td>
</tr>
</tbody>
</table>
## Features comparison (2/2)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Tool 1</th>
<th>Tool 2</th>
<th>Tool 3</th>
<th>Tool 4</th>
<th>Tool 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
</tr>
<tr>
<td>Locating motifs</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Common sequences</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Cleaning bases</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transcription</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Translation</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Filtering by size</td>
<td>Indirect</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Renaming header</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
</tr>
<tr>
<td>Other features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross-platform</td>
<td>Yes</td>
<td>Partly</td>
<td>Partly</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Reading STDIN</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>Yes</td>
</tr>
<tr>
<td>Reading gzipped file</td>
<td>Yes</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
</tr>
<tr>
<td>Writing gzip file</td>
<td>Yes</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Benchmark

- Datasets

<table>
<thead>
<tr>
<th>file</th>
<th>seq_format</th>
<th>seq_type</th>
<th>num_seq</th>
<th>min_len</th>
<th>avg_len</th>
<th>max_len</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset_A.fa</td>
<td>FASTA</td>
<td>DNA</td>
<td>67,748</td>
<td>56</td>
<td>41,442.5</td>
<td>5,976,145</td>
</tr>
<tr>
<td>dataset_B.fa</td>
<td>FASTA</td>
<td>DNA</td>
<td>194</td>
<td>970</td>
<td>15,978,096.5</td>
<td>248,956,422</td>
</tr>
<tr>
<td>dataset_C.fq</td>
<td>FASTQ</td>
<td>DNA</td>
<td>9,186,045</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

- dataset_A.fa - large number of short sequences, 2.7G
- dataset_B.fa - small number of large sequences, 2.9G
- dataset_C.fq – Illumina single end reads (SE100), 2.2G
FASTA/Q Parsing Speed

**Graph:**
- **Title:** FASTA/Q Parsing
- **Axes:**
  - Y-axis: Time (s)
  - X-axis: Peak Memory (MB)
- **Datasets:**
  - dataset_A.fa
  - dataset_B.fa
  - dataset_C.fq
- **Tools:**
  - seqkit
  - seqtk

The graph compares the parsing speed and peak memory usage of different tools and datasets.
Five common manipulations (FASTA)

A) Reverse complement

B) Searching by ID list

C) Sampling by number

D) Removing duplicates by seq

E) Subsequence with BED file

Datasets
- dataset_A.fa
- dataset_B.fa

Tools
- biogo
- fasta_utilities
- seqtk
- seqkit
- seqmagick
- seqtk
Five common manipulations (FASTQ)

B) Searching by ID list

C) Sampling by number

D) Removing duplicates by seq

Datasets
- dataset_C.fq

Tools
- fasta_utilities
- seqkit
- seqmagick
- seqtk
Test of multiple threads

A) Reverse complement

B) Searching by ID list

C) Sampling by number

D) Removing duplicates by seq

E) Subsequence with BED file

Datasets
- dataset_A.fa
- dataset_B.fa
- dataset_C.fq

Threads
- 1
- 2
- 3
- 4
Tests on different file sizes

Files are generated by replicating Human genome chr1 for N times.
Limitations

- Most of the subcommands were designed to handle common manipulations of FASTA and FASTQ. Some manipulations of FASTQ like trimming were not included.
- Three types of file formats interconversion, including FASTQ-FASTA, and FASTA/Q-tabular format were supported. Other formats like BAM/SAM could be converted to FASTQ by others tools.
Examples
Quick Inspection

- seqkit stat

```bash
$ memusg -t seqkit stat dataset_*

<table>
<thead>
<tr>
<th>file</th>
<th>format</th>
<th>type</th>
<th>num_seq</th>
<th>sum_len</th>
<th>min_len</th>
<th>avg_len</th>
<th>max_len</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset_A.fa</td>
<td>FASTA</td>
<td>DNA</td>
<td>67,748</td>
<td>2,807,643,808</td>
<td>56</td>
<td>41,442.5</td>
<td>5,976,145</td>
</tr>
<tr>
<td>dataset_B.fa</td>
<td>FASTA</td>
<td>DNA</td>
<td>194</td>
<td>3,099,750,718</td>
<td>970</td>
<td>15,978,096.5</td>
<td>248,956,422</td>
</tr>
<tr>
<td>dataset_C.fq</td>
<td>FASTQ</td>
<td>DNA</td>
<td>9,186,045</td>
<td>918,604,500</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

elapsed time: 23.068s
peak rss: 905.03 MB
Basic manipulations

head

```bash
$ seqkit head -n 1 dataset_A.fa | more
>ADMP01000001 Bacteroides xylanisolvens SD CC 2a contig00137, whole genome shotgun sequence. [Bacteroides xylanisolvens SD CC 2a]
GCAATGGACCCTTAAACCTACTATATAGTCTATATATAATAATAACTCTCTGTTTAACCATA
ACAATGCGACAGACTTAAATCCCATATAAATAAAAAACGAGCTAATTCTTTTTATAGA
ACCACCTCGCCACCGCATCAAGAAAGCCTAAGGCTTTTCCATAGTACCTCAATTTTCTG
```

rna2dna, reverse complement sequence

```bash
$ seqkit head -n 1 hairpin.fa | seqkit seq --rna2dna |
$ seqkit seq -r -p
>cel-let-7 MI0000001 Caenorhabditis elegans let-7 stem-loop
TCGAAGAGTTCTGTCTCCGGTAAGGTAGAAAATTGCATAGTTCACCGGTGGTAATATTCC
AAACTATACAACCTACTACCTCACCGGATCCACAGTGTA
```

length, GC content

```bash
$ seqkit fx2tab -n -i -H -l -g -B AU hairpin.fa | csvtk -C $ -t pretty | more
#name    seq   qual   length   GC      AU
cel-let-7  99   43.43   56.57
cel-lin-4  94   54.26   45.74
cel-mir-1  96   40.62   59.38
```
Searching by ID list (grep)

$ seqkit head -n 3 dataset_A.fa | seqkit seq -n  
ADMP01000001 Bacteroides xylanisolvens SD CC 2a contig00137, whole genome shotgun sequence. [Bacteroides xylanisolvens SD CC 2a]  
ADMP01000002 Bacteroides xylanisolvens SD CC 2a contig00035, whole genome shotgun sequence. [Bacteroides xylanisolvens SD CC 2a]  
ADMP01000003 Bacteroides xylanisolvens SD CC 2a contig00317, whole genome shotgun sequence. [Bacteroides xylanisolvens SD CC 2a]

$ head -n 3 ids_A.txt  
ACTX01000021  
ADTR01000418  
ACAB02000059  
$ wc -l ids_A.txt  
20138 ids_A.txt

$ memusg -t seqkit grep -f ids_A.txt dataset_A.fa > t  
elapsed time: 3.115s  
peak rss: 46.21 MB

$ seqkit stat t  
file format type num_seqs   sum_len    min_len  avg_len   max_len  
t   FASTA   DNA    20,139  799,917,174    56     39,719.8  4,876,800

$ seqkit seq -n -i t | sort | uniq | wc -l  
20138
Removing duplicated sequences (rmdup)

$ memusg -t seqkit rmdup dataset_A.fa > /dev/null
[INFO] 2 duplicated records removed
elapsed time: 3.888s
peak rss: 57.39 MB

$ memusg -t seqkit rmdup -s dataset_A.fa > /dev/null
[INFO] 12351 duplicated records removed
elapsed time: 5.901s
peak rss: 2.68 GB

$ memusg -t seqkit rmdup -s -m dataset_A.fa > /dev/null
[INFO] 12351 duplicated records removed
elapsed time: 8.272s
peak rss: 59.68 MB

$ memusg -t seqkit rmdup -s dataset_C.fq > /dev/null
[INFO] 501303 duplicated records removed
elapsed time: 33.362s
peak rss: 2.79 GB

[shenwei@Riolinux fakit_benchmark]$ memusg -t seqkit rmdup -s -m dataset_C.fq > /dev/null
[INFO] 501303 duplicated records removed
elapsed time: 48.056s
peak rss: 1.83 GB
fx2tab/tab2fx

Sorting by sequence length

```bash
$ seqkit fx2tab -l dataset_A.fa | sort -t"\t" -e \"\t\" -k 4,4n | seqkit tab2fx > t
```

```bash
$ seqkit head -n 5 t | seqkit fx2tab -n -i -l -H | csvtk -C $ -t pretty
#name          seq   qual   length
ADTK01000030                179
ADTK01000029                292
ACTI01000017                299
GG771689                    378
GG771707                    404
```

```bash
$ seqkit head -n 1000 dataset_A.fa | seqkit sort -l | seqkit head -n 5 | seqkit fx2tab -n -i -l -H | csvtk -C $ -t pretty
#name          seq   qual   length
ADTK01000030                179
ADTK01000029                292
ACTI01000017                299
GG771689                    378
GG771707                    404
```
Locating motifs

$ seqkit head -n 1 dataset_A.fa | seqkit locate -f p.fa | csvtk -t pretty -W 25 | more

<table>
<thead>
<tr>
<th>seqID</th>
<th>patternName</th>
<th>pattern</th>
<th>strand</th>
<th>start</th>
<th>end</th>
<th>matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMP01000001</td>
<td>orf</td>
<td>A[TU]G(?:.{3})+?[TU](?:AG</td>
<td>+</td>
<td>3</td>
<td>14</td>
<td>ATGGACCCTTAA</td>
</tr>
<tr>
<td>ADMP01000001</td>
<td>orf</td>
<td>A[TU]G(?:.{3})+?[TU](?:AG</td>
<td>+</td>
<td>64</td>
<td>78</td>
<td>ATGCGACAGACTTAA</td>
</tr>
<tr>
<td>ADMP01000001</td>
<td>orf</td>
<td>A[TU]G(?:.{3})+?[TU](?:AG</td>
<td>+</td>
<td>496</td>
<td>726</td>
<td>ATGTATTTGCAATTTTTGTCTAATA</td>
</tr>
</tbody>
</table>

sliding → locate → plot

Application
## Others functions

### Sequence and subsequence
- **seq** transform sequences (reverse, complement, extract ID...)
- **subseq** get subsequences by region/gtf/bed, including flanking sequences
- **sliding** sliding sequences, circular genome supported
- **stat** simple statistics of FASTA files
- **faidx** create FASTA index file

### Format conversion
- **fx2tab** covert FASTA/Q to tabular format (and length/GC content/GC skew)
- **tab2fx** covert tabular format to FASTA/Q format
- **fq2fa** covert FASTQ to FASTA

### Searching
- **grep** search sequences by pattern(s) of name or sequence motifs
- **locate** locate subsequences/motifs

### Set operations
- **rmdup** remove duplicated sequences by id/name/sequence
- **common** find common sequences of multiple files by id/name/sequence
- **split** split sequences into files by id/seq region/size/parts
- **sample** sample sequences by number or proportion
- **head** print first N FASTA/Q records

### Edit
- **replace** replace name/sequence by regular expression
- **rename** rename duplicated IDs

### Ordering
- **shuffle** shuffle sequences
- **sort** sort sequences by id/name/sequence

### Misc
- **version** print version information
Acknowledgements

- Lei Zhang (Github ID: jameslz, Weibo: @bitslife) for testing of SeqKit
- Jim Hester, author of fasta_utilities, for advice on early performance improvement of FASTA parsing
- Brian Bushnell, author of BBMaps, for advice on naming of SeqKit and adding accuracy evaluation in benchmarks.
Take-home message

- Version control.
  - Use git, Github
- Documents.
  - Annotation for variable and functions. Use lint tools to check this.
  - Build a project website
- Tests.
  - Unit tests and function tests
- Automation.
  - Packing binaries and testing
- Optimization never ends!

Top 10 metrics for life science software good practices
More utilities

- **csvtk** – Another cross-platform, efficient and practical CSV/TSV Toolkit
- **easy_qsub** – Easily submitting multiple PBS jobs or running local jobs in parallel
- **bio_scripts** – Practical, reusable scripts for Bioinformatics

More:
- [http://bioinf.shenwei.me/](http://bioinf.shenwei.me/)
- [https://github.com/shenwei356](https://github.com/shenwei356)
Just try it

http://bioinf.shenwei.me/seqkit/