J. R. Almeida, A. J. Pinho, J. L. Oliveira, D. Pratas

GTO 2: The genomics-proteomics toolkit

Contents

List of Tables

List of Figures

Preface

License

This document was written in RMarkdown[1] using the bookdown[2] package.

[1] https://rmarkdown.rstudio.com
[2] https://bookdown.org

1

Introduction

Recent advances in DNA sequencing, specifically in next-generation sequencing (NGS), revolutionised the field of genomics, making possible the generation of large amounts of sequencing data very rapidly and at substantially low cost(Mardis, 2017). This new technology also brought with it several challenges, namely in what concerns the analysis, storage, and transmission of the generated sequences(Brouwer et al., 2016, Liu et al. (2012)). As a consequence, several specialised tools were developed throughout the years in order to deal with these challenges.

Firstly, the storage of the raw data generated by NGS experiments is possible by using several file formats, the FASTQ and FASTA are the most commonly used(Zhang, 2016). FASTQ is an extension of the FASTA format, that besides the nucleotide sequence, also stores associated per base quality score and it is considered the standard format for sequencing data storage and exchange(Cock et al., 2009).

Regarding the analysis and manipulation of these sequencing data files many software applications emerged, including **fqtools**(Droop, 2016), **FASTX-Toolkit**(Gordon et al., 2010), **GALAXY**(Afgan et al., 2018), **GATK**(DePristo et al., 2011), **MEGA**(Kumar et al., 2016), **SeqKit**(Shen et al., 2016), among others. **Fqtools** is a suite of tools to view, manipulate and summarise FASTQ data. This software also identifies invalid FASTQ files(Droop, 2016). **GALAXY**, in its turn, is an open, web-based scientific platform for analysing genomic data(Goecks et al., 2010). This platform integrates several specialised sets of tools, e.g. for manipulating FASTQ files(Blankenberg et al., 2010). **FASTX-Toolkit** is a collection of command-line tools to process FASTA and FASTQ files. This toolkit is available in two forms: as a command-line, or integrated into the web-based platform **GALAXY**(Gordon et al., 2010). **SeqKit** is another toolkit used to process FASTA and FASTQ files and is available for all major operating systems(Shen et al., 2016). The Genome Analysis Toolkit (**GATK**) was designed as a structured programming framework

to simplify the development of analysis tools. However, nowadays, it is a suite of tools focused on variant discovering and genotyping(Van der Auwera et al., 2013). More towards the evolutionary perspectives, Molecular Evolutionary Genetics Analysis (**MEGA**) software provides tools to analyse DNA and protein sequences statistically(Tamura et al., 2011). Several of these frameworks lack on variety, namely the ability to perform multiple tasks using only one toolkit.

Compression is another important aspect when dealing with high-throughput sequencing data, as it reduces storage space and accelerates data transmission. A survey on DNA compressors and amino acid sequence compression can be found in(Hosseini et al., 2016). Currently, the DNA sequence compressors HiRGC(Liu et al., 2017), iDoComp(Ochoa et al., 2014), GeCo(Pratas et al., 2016), and GDC(Deorowicz et al., 2015) are considered to have the best performance(Hernaez et al., 2019). Of these four approaches, GeCo is the only one that can be used for reference-free and reference-based compression. Furthermore, GeCo can be used as an analysis tool to determine absolute measures for many distance computations and local measures(Pratas et al., 2016).

Amino acid sequences are known to be very hard to compress(Nalbantoglu et al., 2010), however, Hosseini et al.(Hosseini et al., 2019) recently developed AC, a state-of-the-art for lossless amino acid sequence compression. In(Pratas et al., 2018) the authors compared the performance of AC, in terms of bit-rate, to several general-purpose lossless compressors and several protein compressors, using different proteomes. They concluded that in average AC provides the best bit-rates.

Another relevant subject is genomic data simulation. Read simulations tools are fundamental for the development, testing and evaluation of methods and computational tools(Huang et al., 2011, price2017simulome). Despite the availability of a large number of real sequence reads, read simulation data is necessary due to the inability to know the ground truth of real data(Baruzzo et al., 2017). Escalona *et al.*(Escalona et al., 2016), recently, reviewed 23 NGS simulation tools. XS(Pratas et al., 2014), a FASTQ read simulation tool, stands out in relation to the other 22 simulation tools because it is the only one that does not need a reference sequence. Furthermore, XS is the only open-source tool for simulation of FASTQ reads produced by the four most

used sequencing machines, Roche-454, Illumina, ABI SOLiD and Ion Torrent.

Although a large number of tools are available for analysing, compressing, and simulation, these tools are specialised in only a specific task. Besides, in many cases the output of one tool cannot be used directly as input for another tool, e.g. the output of a simulation tool cannot always be used directly as input for an analysis tool. Thus, unique software that includes several specialised tools is necessary.

In this document, we describe **GTO2**, a complete toolkit for genomics and proteomics, namely for FASTQ, FASTA and SEQ formats, with many complementary tools. The toolkit is for Unix-based systems, built for ultra-fast computations. **GTO2** supports pipes for easy integration with the sub-programs belonging to **GTO2** as well as external tools. **GTO2** works as **LEGOs**, since it allows the construction of multiple pipelines with many combinations.

GTO2 includes tools for information display, randomisation, edition, conversion, extraction, search, calculation, compression, simulation and visualisation. **GTO2** is prepared to deal with very large datasets, typically in the scale of Gigabytes or Terabytes (but not limited). The complete toolkit is an optimised command-line version, using the prefix `gto2_` followed by the suffix with the respective name of the program. **GTO2** is implemented in **C** language and it is available, under the MIT license, at https://github.com/cobilab/gto2

1.1 Installation

To install **GTO2** through the GitHub repository:

```
git clone https://github.com/cobilab/gto2.git
cd gto2/src/
make
```

Or by installing them directly using the Cobilab channel from Conda:

```
conda install -c cobilab gto2 -y
```

1.2 Testing

The examples provided in this document are available in the repository. Therefore, each example can be easily reproduced, which it will also test and validate each tool. To replicate those tests, it can be done in two different ways:

- Running one test for a specific tool:
  - cd gto2/tester/gto2_{tool}
  - sh runExample.sh
- Running the batch of tests for all the tools:
  - cd gto2/tester/
  - sh runAllTests.sh

Some of these tests require internet connection to download external files and it will create new files.

1.3 Execution control

The quality control in Unix/Linux pipelines using GTO's tools is made in three ways:

- Input verification: where the tools verify the format of the input file;
- Stderr logs: Some execution errors are directly sent for the stderr channel.
- Scripting validation: In complex pipelines, the verification of all the tools in the pipeline were executed properly, it is used the PIPESTATUS variable, e.g.:

```
gto2_fa_rand_extra_chars < input.fa | \
gto2_fa_to_seq > output.seq
echo "${PIPESTATUS[0]} ${PIPESTATUS[1]}"
0 0
```

2

FASTA Tools

2.1 gto2_fa_to_fq

to do

3

FASTQ Tools

The toolkit has a set of tools dedicated to manipulating FASTQ files. Some of these tools allow the data conversion to/from different formats, i. e., there are tools designed to convert a FASTQ file into a sequence or a FASTA/Multi-FASTA format, or converting DNA in some of those formats to FASTQ.

There are also tools for data manipulation in this format, which are designed to exclude 'N', remove low quality scored reads, following different metrics and randomize DNA sequences. Succeeding the manipulation, it is also possible to perform analyses over these files, simulations and mutations. The current available tools for FASTQ format analysis and manipulation include:

- **gto2_fq_to_fa**: to convert a FASTQ file format to a pseudo FASTA file.
- **gto2_fq_to_mfa**: to convert a FASTQ file format to a pseudo Multi-FASTA file.
- **gto2_fq_exclude_n**: to discard the FASTQ reads with the minimum number of "N" symbols.
- **gto2_fq_extract_quality_scores**: to extract all the quality-scores from FASTQ reads.
- **gto2_fq_info**: to analyse the basic information of FASTQ file format.
- **gto2_fq_maximum_read_size**: to filter the FASTQ reads with the length higher than the value defined.
- **gto2_fq_minimum_quality_score**: to discard reads with average quality-score below of the defined.
- **gto2_fq_minimum_read_size**: to filter the FASTQ reads with the length smaller than the value defined.
- **gto2_fq_rand_extra_chars**: to substitue in the FASTQ files, the DNA sequence the outside ACGT chars by random ACGT symbols.

- **gto2_fq_from_seq**: to convert a genomic sequence to pseudo FASTQ file format.
- **gto2_fq_mutate**: to create a synthetic mutation of a FASTQ file given specific rates of mutations, deletions and additions.
- **gto2_fq_split**: to split Paired End files according to the direction of the strand ('/1' or '/2').
- **gto2_fq_pack**: to package each FASTQ read in a single line.
- **gto2_fq_unpack**: to unpack the FASTQ reads packaged using the **gto2_fq_pack** tool.
- **gto2_fq_quality_score_info**: to analyse the quality-scores of a FASTQ file.
- **gto2_fq_quality_score_min**: to analyse the minimal quality-scores of a FASTQ file.
- **gto2_fq_quality_score_max**: to analyse the maximal quality-scores of a FASTQ file.
- **gto2_fq_cut**: to cut read sequences in a FASTQ file.
- **gto2_fq_minimum_local_quality_score_forward**: to filter the reads considering the quality score average of a defined window size of bases.
- **gto2_fq_minimum_local_quality_score_reverse**: to filter the reverse reads, considering the average window size score defined by the bases.
- **gto2_fq_xs**: a skilled FASTQ read simulation tool, flexible, portable and tunable in terms of sequence complexity.
- **gto2_fq_complement**: to replace the ACGT bases with their complements in a FASTQ file format.
- **gto2_fq_reverse**: to reverse the ACGT bases order for each read in a FASTQ file format.
- **gto2_fq_variation_map**: to identify the variation that occours in the sequences relative to the reads or a set of reads.
- **gto2_fq_variation_filter**: to filter and segments the regions of singularity from the output of **gto2_fq_variation_map**.
- **gto2_fq_variation_visual**: to depict the regions of singularity using the output from **gto2_fq_variation_filter** into an SVG image.
- **gto2_fq_metagenomics**: to measure the similarity between any FASTQ file, independently from the size, against any multi-FASTA database.

3.1 Program gto2_fq_to_fa

The **gto2_fq_to_fa** converts a FASTQ file format to a pseudo FASTA file. However, this tool does not align the sequence, instead, it extracts the sequence and adds a pseudo-header.

For help type:

```
./gto2_fq_to_fa -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_to_fa** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_to_fa [options] [[--] args]
   or: ./gto2_fq_to_fa [options]

It converts a FASTQ file format to a pseudo FASTA file.
It does NOT align the sequence.
It extracts the sequence and adds a pseudo header.

    -h, --help            show this help message and exit

Basic options
    < input.fastq         Input FASTQ file format (stdin)
    > output.fasta        Output FASTA file format (stdout)

Example: ./gto2_fq_to_fa < input.fastq > output.fasta
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
```

```
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_to_fa** program is a FASTA file. Using the input above, an output example of this is the following:

```
> Computed with Fastq2Fasta
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
```

3.2 Program gto2_fq_to_mfa

The **gto2_fq_to_mfa** converts a FASTQ file format to a pseudo Multi-FASTA file. However, this tool does not align the sequence, instead, it extracts the sequence and adds a pseudo header.

For help type:

```
./gto2_fq_to_mfa -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_to_mfa** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_to_mfa [options] [[--] args]
   or: ./gto2_fq_to_mfa [options]

It converts a FASTQ file format to a pseudo Multi-FASTA file.
It does NOT align the sequence.
It extracts the sequence and adds each header in a Multi-FASTA format.

    -h, --help            show this help message and exit

Basic options
    < input.fastq         Input FASTQ file format (stdin)
    > output.mfasta       Output Multi-FASTA file format (stdout)

Example: ./gto2_fq_to_mfa < input.fastq > output.mfasta
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_to_mfa** program is a Multi-FASTA file. Using the input above, an output example of this is the following:

```
>SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
>SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
```

3.3 Program gto2_fq_exclude_n

The **gto2_fq_exclude_n** discards the FASTQ reads with the minimum number of ''N" symbols, and it will erase the second header (after +), if presented.

For help type:

```
./gto2_fq_exclude_n -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_exclude_n** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_exclude_n [options] [[--] args]
   or: ./gto2_fq_exclude_n [options]

It discards the FASTQ reads with the minimum number of "N"
symbols.
If present, it will erase the second header (after +).

    -h, --help            show this help message and exit

Basic options
    -m, --max=<int>       The maximum of of "N" symbols in
                          the read
    < input.fastq         Input FASTQ file format (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_exclude_n -m <max> < input.fastq >
output.fastq
```

```
Console output example :
<FASTQ non-filtered reads>
Total reads    : value
Filtered reads : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_exclude_n** program is a set of all the filtered FASTQ reads, followed by the execution report. The execution report only appears in the console.

Using the input above with the max value as 5, an output example for this is the following:

```
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
Total reads    : 2
Filtered reads : 1
```

3.4 Program gto2_fq_extract_quality_scores

The **gto2_fq_extract_quality_scores** extracts all the quality-scores from FASTQ reads.

For help type:

```
./gto2_fq_extract_quality_scores -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_extract_quality_scores** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_extract_quality_scores [options] [[--]args]
   or: ./gto2_fq_extract_quality_scores [options]

It extracts all the quality-scores from FASTQ reads.

    -h, --help            show this help message and exit

Basic options
    < input.fastq         Input FASTQ file format (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_extract_quality_scores < input.fastq >
output.fastq

Console output example:
<FASTQ quality scores>
Total reads          : value
Total Quality-Scores : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
```

```
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_extract_quality_scores** program is a set of all the quality scores from the FASTQ reads, followed by the execution report. The execution report only appears in the console. Using the input above, an output example of this is the following:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIDIII
IIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
Total reads          : 2
Total Quality-Scores : 144
```

3.5 Program gto2_fq_info

The **gto2_fq_info** analyses the basic information of FASTQ file format.

For help type:

```
./gto2_fq_info -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_info** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_info [options] [[--] args]
   or: ./gto2_fq_info [options]

It analyses the basic information of FASTQ file format.

    -h, --help            show this help message and exit

Basic options
    < input.fastq         Input FASTQ file format (stdin)
    > output              Output read information (stdout)

Example: ./gto2_fq_info < input.fastq > output

Output example:
Total reads     : value
Max read length : value
Min read length : value
Min QS value    : value
Max QS value    : value
QS range        : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_info** program is a set of information related to the file read. Using the input above, an output example of this is the following:

```
Total reads     : 2
Max read length : 72
```

```
Min read length : 72
Min QS value    : 41
Max QS value    : 73
QS range        : 33
```

3.6 Program gto2_fq_maximum_read_size

The **gto2_fq_maximum_read_size** filters the FASTQ reads with the length higher than the value defined.

For help type:

```
./gto2_fq_maximum_read_size -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_maximum_read_size** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_maximum_read_size [options] [[--] args]
   or: ./gto2_fq_maximum_read_size [options]

It filters the FASTQ reads with the length higher than the
value defined.
If present, it will erase the second header (after +).

    -h, --help            show this help message and exit

Basic options
    -s, --size=<int>      The maximum read length
    < input.fastq         Input FASTQ file format (stdin)
```

```
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_maximum_read_size -s <size> < input.fastq
> output.fastq

Console output example :
<FASTQ non-filtered reads>
Total reads    : value
Filtered reads : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=59
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_maximum_read_size** program is a set of all the filtered FASTQ reads, followed by the execution report. The execution report only appears in the console.

Using the input above with the size values as 59, an output example for this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=59
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDII
Total reads    : 2
Filtered reads : 1
```

3.7 Program gto2_fq_minimum_quality_score

The **gto2_fq_minimum_quality_score** discards reads with average quality-score below of the defined.

For help type:

```
./gto2_fq_minimum_quality_score -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_minimum_quality_score** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_minimum_quality_score [options] [[--] args]
   or: ./gto2_fq_minimum_quality_score [options]

It discards reads with average quality-score below value.

    -h, --help            show this help message and exit

Basic options
    -m, --min=<int>       The minimum average quality-score
                          (Value 25 or 30 is commonly used)
    < input.fastq         Input FASTQ file format (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_minimum_quality_score -m <min> <
input.fastq > output.fastq

Console output example:
<FASTQ non-filtered reads>
```

```
Total reads    : value
Filtered reads : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
54599<>77977==6=?I6IBI::33344235521677999>>>><<<@@A@BBCDGGBFF
```

Output

The output of the **gto2_fq_minimum_quality_score** program is a set of all the filtered FASTQ reads, followed by the execution report. Using the input above with the minimum averge value as 30, an output example of this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
Total reads    : 2
Filtered reads : 1
```

3.8 Program gto2_fq_minimum_read_size

The **gto2_fq_minimum_read_size** filters the FASTQ reads with the length smaller than the value defined.

For help type:

```
./gto2_fq_minimum_read_size -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_minimum_read_size** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_minimum_read_size [options] [[--] args]
   or: ./gto2_fq_minimum_read_size [options]

It filters the FASTQ reads with the length smaller than the
value defined.
If present, it will erase the second header (after +).

    -h, --help            show this help message and exit

Basic options
    -s, --size=<int>      The minimum read length
    < input.fastq         Input FASTQ file format (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_minimum_read_size -s <size> < input.fastq
> output.fastq

Console output example:
<FASTQ non-filtered reads>
Total reads    : value
Filtered reads : value
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=50
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAAC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
```

```
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_minimum_read_size** program is a set of all the filtered FASTQ reads, followed by the execution report. The execution report only appears in the console. Using the input above with the size values as 55, an output example of this is the following:

```
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
Total reads    : 2
Filtered reads : 1
```

3.9 Program gto2_fq_rand_extra_chars

The **gto2_fq_rand_extra_chars** substitutes the outside ACGT chars by random ACGT symbols in the DNA sequence of FASTQ files.

For help type:

```
./gto2_fq_rand_extra_chars -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_rand_extra_chars** program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_rand_extra_chars [options] [[--] args]
   or: ./gto2_fq_rand_extra_chars [options]

It substitues in the FASTQ files, the DNA sequence the
outside ACGT chars by random ACGT symbols.

    -h, --help            show this help message and exit

Basic options
    < input.fastq         Input FASTQ file format (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Example: ./gto2_fq_rand_extra_chars < input.fastq >
output.fastq
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

Output

The output of the **gto2_fq_rand_extra_chars** program is a FASTQ file. Using the input above, an output example of this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GTGTGATGGCCGCTGCCGATGGCGCATAATCCCACCAACATACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGATTGTATTTTAAGAATCTGCAGCAGAAGTCGATGATAATACGCG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGI
```

3.10 Program gto2_fq_from_seq

The **gto2_fq_from_seq** converts a genomic sequence to pseudo FASTQ file format.

For help type:

```
./gto2_fq_from_seq -h
```

In the following subsections, we explain the input and output parameters.

Input parameters

The **gto2_fq_from_seq** program needs two streams for the computation, namely the input and output standard. The input stream is a sequence group file.

The attribution is given according to:

```
Usage: ./gto2_fq_from_seq [options] [[--] args]
   or: ./gto2_fq_from_seq [options]

It converts a genomic sequence to pseudo FASTQ file format.

    -h, --help            show this help message and exit

Basic options
    < input.seq           Input sequence file (stdin)
    > output.fastq        Output FASTQ file format (stdout)

Optional options
    -n, --name=<str>      The read's header
    -l, --lineSize=<int>  The maximum of chars for line

Example: ./gto2_fq_from_seq -l <lineSize> -n <name> <
input.seq > output.fastq
```

An example of such an input file is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCT
GCCCTGCTGCCATTGTCCCCGGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGC
TTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAAGTGGTTTGAGTGGACCTCCG
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAG
```

Output

The output of the **gto2_fq_from_seq** program is a pseudo FASTQ file. An example, using the size line as 60 and the read's header as ''SeqToFastq", for the input, is:

```
@SeqToFastq1
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq2
GCCCTGCTGCCATTGTCCCCGGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq3
TTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAAGTGGTTTGAGTGGACCTCCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq4
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq5
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

3.11 Program gto2_fq_mutate

to do

3.12 Program gto2_fq_split

to do

3.13 Program gto2_fq_pack

to do

3.14 Program gto2_fq_unpack

to do

3.15 Program gto2_fq_quality_score_info

to do

3.16 Program gto2_fq_quality_score_min

to do

3.17 Program gto2_fq_quality_score_max

to do

3.18 Program gto2_fq_cut

to do

3.19 Program gto2_fq_minimum_local_quality_score_forward

to do

3.20 Program gto2_fq_minimum_local_quality_score_reverse

to do

3.21 Program gto2_fq_xs

to do

3.22 Program gto2_fq_complement

to do

3.23 Program gto2_fq_reverse

to do

3.24 Program gto2_fq_variation_map

to do

3.25 Program gto2_fq_variation_filter

to do

3.26 Program gto2_fq_variation_visual

to do

3.27 Program gto2_fq_metagenomics

to do

4

Amino Acid Tools

4.1 gto2_aa

to do

5

Genomic Tools

5.1 gto2_dna

to do

6

General Purpose Tools

6.1 gto2_

to do

Bibliography

Afgan, E., Baker, D., Batut, B., Van Den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., et al. (2018). The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1):W537–W544.

Baruzzo, G., Hayer, K. E., Kim, E. J., Di Camillo, B., FitzGerald, G. A., and Grant, G. R. (2017). Simulation-based comprehensive benchmarking of rna-seq aligners. *Nature methods*, 14(2):135.

Blankenberg, D., Gordon, A., Von Kuster, G., Coraor, N., Taylor, J., Nekrutenko, A., and Team, G. (2010). Manipulation of fastq data with galaxy. *Bioinformatics*, 26(14):1783–1785.

Brouwer, C., Vu, T. D., Zhou, M., Cardinali, G., Welling, M. M., van de Wiele, N., and Robert, V. (2016). Current opportunities and challenges of next generation sequencing (ngs) of dna; determining health and disease. *British Biotechnology Journal*, 13(4).

Cock, P. J., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2009). The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771.

Deorowicz, S., Danek, A., and Niemiec, M. (2015). Gdc 2: Compression of large collections of genomes. *Scientific reports*, 5:11565.

DePristo, M. A., Banks, E., Poplin, R., Garimella, K. V., Maguire, J. R., Hartl, C., Philippakis, A. A., Del Angel, G., Rivas, M. A., Hanna, M., et al. (2011). A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491.

Droop, A. P. (2016). fqtools: an efficient software suite for modern fastq file manipulation. *Bioinformatics*, 32(12):1883–1884.

Escalona, M., Rocha, S., and Posada, D. (2016). A comparison of tools for the simulation of genomic next-generation sequencing data. *Nature Reviews Genetics*, 17(8):459.

Goecks, J., Nekrutenko, A., and Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86.

Gordon, A., Hannon, G., et al. (2010). Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*, 5.

Hernaez, M., Pavlichin, D., Weissman, T., and Ochoa, I. (2019). Genomic data compression. *Annual Review of Biomedical Data Science*, 2.

Hosseini, M., Pratas, D., and Pinho, A. (2016). A survey on data compression methods for biological sequences. *Information*, 7(4):56.

Hosseini, M., Pratas, D., and Pinho, A. J. (2019). Ac: A compression tool for amino acid sequences. *Interdisciplinary Sciences: Computational Life Sciences*, pages 1–9.

Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2011). Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

Kumar, S., Stecher, G., and Tamura, K. (2016). Mega7: molecular evolutionary genetics analysis version 7.0 for bigger datasets. *Molecular biology and evolution*, 33(7):1870–1874.

Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., and Law, M. (2012). Comparison of next-generation sequencing systems. *BioMed Research International*, 2012.

Liu, Y., Peng, H., Wong, L., and Li, J. (2017). High-speed and high-ratio referential genome compression. *Bioinformatics*, 33(21):3364–3372.

Mardis, E. R. (2017). Dna sequencing technologies: 2006–2016. *Nature protocols*, 12(2):213.

Nalbantoglu, Ö., Russell, D., and Sayood, K. (2010). Data compression concepts and algorithms and their applications to bioinformatics. *Entropy*, 12(1):34–52.

Ochoa, I., Hernaez, M., and Weissman, T. (2014). idocomp: a compression scheme for assembled genomes. *Bioinformatics*, 31(5):626–633.

Pratas, D., Hosseini, M., and Pinho, A. J. (2018). Compression of amino acid sequences. In *International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 105–113. Springer.

Pratas, D., Pinho, A. J., and Ferreira, P. J. (2016). Efficient compression of genomic sequences. In *2016 Data Compression Conference (DCC)*, pages 231–240. IEEE.

Pratas, D., Pinho, A. J., and Rodrigues, J. M. (2014). Xs: a fastq read simulator. *BMC research notes*, 7(1):40.

Shen, W., Le, S., Li, Y., and Hu, F. (2016). Seqkit: a cross-platform and ultrafast toolkit for fasta/q file manipulation. *PLoS One*, 11(10):e0163962.

Tamura, K., Peterson, D., Peterson, N., Stecher, G., Nei, M., and Kumar, S. (2011). Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular biology and evolution*, 28(10):2731–2739.

Van der Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., Del Angel, G., Levy-Moonshine, A., Jordan, T., Shakir, K., Roazen, D., Thibault, J., et al. (2013). From fastq data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Current protocols in bioinformatics*, 43(1):11–10.

Zhang, H. (2016). Overview of sequence data formats. In *Statistical Genomics*, pages 3–17. Springer.