

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

---

# ***GTO 2: The genomics-proteomics toolkit***



---

# *Contents*

---

<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>Preface</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Installation . . . . .	3
1.2 Testing . . . . .	4
1.3 Execution control . . . . .	4
<b>2 FASTA Tools</b>	<b>5</b>
2.1 gto2_fa_to_fq . . . . .	5
<b>3 FASTQ Tools</b>	<b>7</b>
3.1 Program gto2_fq_to_fa . . . . .	9
3.2 Program gto2_fq_to_mfa . . . . .	10
3.3 Program gto2_fq_exclude_n . . . . .	12
3.4 Program gto2_fq_extract_quality_scores . . . . .	13
3.5 Program gto2_fq_info . . . . .	15
3.6 Program gto2_fq_maximum_read_size . . . . .	17
3.7 Program gto2_fq_minimum_quality_score . . . . .	19
3.8 Program gto2_fq_minimum_read_size . . . . .	20
3.9 Program gto2_fq_rand_extra_chars . . . . .	22
3.10 Program gto2_fq_from_seq . . . . .	24
3.11 Program gto2_fq_mutate . . . . .	26
3.12 Program gto2_fq_split . . . . .	28
3.13 Program gto2_fq_pack . . . . .	30
3.14 Program gto2_fq_unpack . . . . .	31
3.15 Program gto2_fq_quality_score_info . . . . .	33
3.16 Program gto2_fq_quality_score_min . . . . .	35
3.17 Program gto2_fq_quality_score_max . . . . .	37

3.18	Program gto2_fq_cut . . . . .	38
3.19	Program gto2_fq_minimum_local_quality_score_forward 40	
3.20	Program gto2_fq_minimum_local_quality_score_reverse 42	
3.21	Program gto2_fq_xs . . . . .	44
3.22	Program gto2_fq_complement . . . . .	45
3.23	Program gto2_fq_reverse . . . . .	47
3.24	Program gto2_fq_variation_map . . . . .	48
3.25	Program gto2_fq_variation_filter . . . . .	51
3.26	Program gto2_fq_variation_visual . . . . .	52
3.27	Program gto2_fq_metagenomics . . . . .	53
<b>4</b>	<b>Amino Acid Tools</b>	<b>57</b>
4.1	gto2_aa . . . . .	57
<b>5</b>	<b>Genomic Tools</b>	<b>59</b>
5.1	gto2_dna . . . . .	59
<b>6</b>	<b>General Purpose Tools</b>	<b>61</b>
6.1	gto2_ . . . . .	61

---

## *List of Tables*

---



---

## *List of Figures*

---

3.1 Execution plot of the variation visual tool using the previous input. . . . .	53
---	----





---

# *Preface*

---

---

## License

This document was written in RMarkdown<sup>1</sup> using the bookdown<sup>2</sup> package.

---

<sup>1</sup><https://rmarkdown.rstudio.com>

<sup>2</sup><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 substitute 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 occurs 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
```

```

+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG

```

---

## 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:

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.

## Basic options

Example: `./gto2fq_to_mfa < input.fastq > output.mfasta`

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIGI
```

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:

>SRRO01666.1 071112\_SLXA-EAS1\_s\_7:5:1:817:345 length=60  
GGGTGATGGCCGCTGCCGATGGCGTCAAAATCCCACCAAGTTACCCTTAACAACTTAAGGG  
>SRRO01666.2 071112\_SLXA-EAS1\_s\_7:5:1:801:338 length=60  
GTTCAAGGGATACGACGTTTGTATTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG

### 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
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI

```

## 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
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
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  
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACA ACTTAAGGG  
+  
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIDIII  
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
```



```
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIIGI
```

## 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:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIIGI
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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCGTTAACAACCTTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCGTTAACAACCTTAAGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDII
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>
```

An example of such an input file is:

## Output

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
Total reads      : 2
Filtered reads   : 1
```

### 3.8 Program gto2\_fq\_minimum\_read\_size

For help type:

```
./gto2fq_minimum_read_size -h
```

## Input parameters

The attribution is given according to:

If present, it will erase the second header (after +).

## Basic options

```

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

```

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=50  
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAAC  
+  
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIII  
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60  
GTTCAGGGATAACGACGTTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACCGG
```

```
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## 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
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
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:





### 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  
GCCCTGCTGCCATTGTCCCCGGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGC  
TTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAAGTGGTTTGAGTGGACCTCCG  
GGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC  
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAAGTCTCTCTGGAAG

## 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:

[illegible]

### 3.11 Program `gto2_fq_mutate`

The `gto2_fq_mutate` creates a synthetic mutation of a FASTQ file given specific rates of mutations, deletions and additions. All these parameters are defined by the user, and their are optional.

For help type:

```
./gto2_fq_mutate -h
```

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

#### Input parameters

The `gto2_fq_mutate` program needs two streams for the computation, namely the input and output standard. However, optional settings can be supplied too, such as the starting point to the random generator, and the edition, deletion and insertion rates. Also, the user can choose to use the ACGTN alphabet in the synthetic mutation. The input stream is a FASTQ File.

The attribution is given according to:

```
Usage: ./gto2_fq_mutate [options] [--] args
      or: ./gto2_fq_mutate [options]
```

Creates a synthetic mutation of a FASTQ file given specific rates of mutations, deletions and additions.

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

#### Basic options

```
< input.fasta   Input FASTQ file format (stdin)
> output.fasta  Output FASTQ file format (stdout)
```

#### Optional

```
-s              Starting point to the random generator
-m             Defines the mutation rate (default 0.0)
```



### 3.12 Program gto2\_fq\_split

The **gto2\_fq\_split** splits Paired End files according to the direction of the strand ('/1' or '/2'). It writes by default singleton reads as forward stands.

For help type:

```
./gto2_fq_split -h
```

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

#### Input parameters

The **gto2\_fq\_split** program needs a stream for the computation, namely the input standard. The input stream is a FASTQ file.

The attribution is given according to:

```
Usage: ./gto2_fq_split [options] [--] args
       or: ./gto2_fq_split [options]
```

It writes by default singleton reads as forward stands.

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

#### Basic options

```
-f, --forward=<str>  Output forward file
-r, --reverse=<str>  Output reverse file
< input.fastq        Input FASTQ file format (stdin)
> output              Output read information (stdout)
```

```
Example: ./gto2_fq_split -f <output_forward.fastq>
-r <output_reverse.fastq> < input.fastq > output
```

Output example :

```
Total reads      : value
Singleton reads  : value
```



### 3.13 Program gto2\_fq\_pack

The **gto2\_fq\_pack** packages each FASTQ read in a single line. It can show the read score first or the dna sequence, depending on the execution mode.

For help type:

```
./gto2_fq_pack -h
```

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

#### Input parameters

The **gto2\_fq\_pack** 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_pack [options] [--] args]
or: ./gto2_fq_pack [options]
```

It packages each FASTQ read in a single line.

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

#### Basic options

```
< input.fastq      Input FASTQ file format (stdin)
> output.fastqpack Output packaged FASTQ file format
                    (stdout)
```

#### Optional

```
-s, --scores        When active, the application show
                    the scores first
```

```
Example: ./gto2_fq_pack -s < input.fastq > output.fastqpack
```



An example of such an input file is:

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

## Output

The output of the `gto2_fq_pack` program is a packaged FASTQ file. Using the input above, an output example of this is the following:

```
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACTTAAGGG
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60+ 0
NTTCAGGGATACGACGNTTGTATTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60+ 1
```

Another example for the same input, but using the scores first (flag ‘s’), is:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACTTAAGGG
SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60+ 0
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
NTTCAGGGATACGACGNTTGTATTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60+ 1
```

---

## 3.14 Program `gto2_fq_unpack`

The `gto2_fq_unpack` unpacks the FASTQ reads packaged using the `gto2_fq_pack` tool.

```
./gto2_fq_unpack -h
```

## Input parameters

```
Usage: ./gto2_fq_unpack [options] [--] args]
       or: ./gto2_fq_unpack [options]
```

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

```
-s, --scores      When active, the application show
                  the scores first
```

GNNATGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGG  
 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII  
 SRR001666.1 071112\_SLXA-EAS1\_s\_7:5:1:817:345 length=60+ 0  
 NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG  
 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI  
 SRR001666.2 071112\_SLXA-EAS1\_s\_7:5:1:801:338 length=60+ 1

## Output

The output of the **gto2\_fq\_unpack** 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
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCGTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIIGI
```

---

## 3.15 Program `gto2_fq_quality_score_info`

The **gto2\_fq\_quality\_score\_info** analyses the quality-scores of a FASTQ file.

For help type:

```
./gto2_fq_quality_score_info -h
```

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

### Input parameters

The **gto2\_fq\_quality\_score\_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_quality_score_info [options] [--] args]
       or: ./gto2_fq_quality_score_info [options]
```

It analyses the quality-scores of a FASTQ file.

```

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

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

Optional
    -m, --max=<int>      The lenght of the maximum window

Example: ./gto2_fq_quality_score_info -m <max> < 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:

```

@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=60 1
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=60 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI

```

## Output

The output of the **gto2\_fq\_quality\_score\_info** program is a set of information related to the file read. Using the input above with the max window value as 30, an output example of this is the following:

```

Total reads      : 2
Max read length : 60

```

```

Min read length : 60
Min QS value    : 54
Max QS value    : 73
QS range        : 20
 1  ...  24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
---+ ... +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
73  ...  73 73 73 73 73 73 73 65 73 62 65 69 70 73 73 73 73
                                     *   * * * *
                                     *   * * * *
                                     *   * * * *
                                     *   * * *
                                     *   * *
                                     *   * *
                                     *   * *
                                     *   * *
                                     *   * *
                                     *
                                     *
                                     *

```

---

### 3.16 Program `gto2_fq_quality_score_min`

The `gto2_fq_quality_score_min` analyses the minimal quality-scores of a FASTQ file.

For help type:

```
./gto2_fq_quality_score_min -h
```

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

#### Input parameters

The `gto2_fq_quality_score_min` 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_quality_score_min [options] [--] args]
or: ./gto2_fq_quality_score_min [options]
```

It analyses the minimal quality-scores of a FASTQ file.

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

#### Basic options

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

#### Optional

```
-m, --max=<int>    The maximum window length (default 40)
```

```
Example: ./gto2_fq_quality_score_min -m <max> < input.fastq
> output
```

An example of such an input file is:

```
@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=60 1
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=60 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## Output

The output of the **gto2\_fq\_quality\_score\_min** program is a set of information related to the file read, considering the minimum quality scores. Using the input above with the max window value as 20, an output example of this is the following:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
--+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73
```

### 3.17 Program `gto2_fq_quality_score_max`

The `gto2_fq_quality_score_max` analyses the maximal quality-scores of a FASTQ file.

For help type:

```
./gto2_fq_quality_score_max -h
```

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

#### Input parameters

The `gto2_fq_quality_score_max` 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_quality_score_max [options] [--] args]
or: ./gto2_fq_quality_score_max [options]
```

It analyses the maximal quality-scores of a FASTQ file.

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

#### Basic options

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

#### Optional

```
-m, --max=<int>    The maximum window length (default 40)
```

```
Example: ./gto2_fq_quality_score_max -m <max>
< input.fastq > output
```

An example of such an input file is:

```
@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=60 1
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=60 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## Output

The output of the **gto2\_fq\_quality\_score\_max** program is a set of information related to the file read, considering the maximal quality scores. Using the input above with the max window value as 20, an output example of this is the following:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
--+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73
```

---

### 3.18 Program gto2\_fq\_cut

The **gto2\_fq\_cut** cuts read sequences in a FASTQ file. It requires that the initial and end positions for the cut.

For help type:

```
./gto2_fq_cut -h
```

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



## Input parameters

The **gto2\_fq\_cut** 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_cut [options] [--] args]
       or: ./gto2_fq_cut [options]
```

It cuts read sequences in a FASTQ file.

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

### Basic options

```
-i, --initial=<int>  Starting position to the cut
-e, --end=<int>      Ending position to the cut
< input.fastq       Input FASTQ file format (stdin)
> output.fastq       Output FASTQ file format (stdout)
```

```
Example: ./gto2_fq_cut -i <initial> -e <end> < 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## Output

The output of the **gto2\_fq\_cut** program is a FASTQ file cut. Using the initial value as 10 and the end value as 30, an example of this input, is the following:

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

---

### 3.19 Program `gto2_fq_minimum_local_quality_score_forward`

The `gto2_fq_minimum_local_quality_score_forward` filters the reads considering the quality score average of a defined window size of bases.

For help type:

```
./gto2_fq_minimum_local_quality_score_forward -h
```

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

#### Input parameters

The `gto2_fq_minimum_local_quality_score_forward` 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_local_quality_score_forward [options] [--] args]
or: ./gto2_fq_minimum_local_quality_score_forward [options]
```

It filters the reads considering the quality score average of a defined window size of bases.

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

**Basic options**

<b>-k</b>	The window size of bases (default 5)
<b>-w</b>	The minimum average of quality score (default 25)
<b>-m</b>	The minimum value of the quality score (default 33)
<b>&lt; input.fastq</b>	Input FASTQ file format (stdin)
<b>&gt; output.fastq</b>	Output FASTQ file format (stdout)

Example: `./gto2_fq_minimum_local_quality_score_forward`

`-k <windowsize> -w <minavg> -m <minqs>`

`< input.fastq > output.fastq`

Console output example:

Minimum QS : value

<FASTQ output>

Total reads : value

Trimmed reads : value

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIII-I)8IIIIIIIIIIIIIIIIII
```

**Output**

The output of the `gto2_fq_minimum_local_quality_score_forward` program is a FASTQ file with the reads filtered following a quality score average of a defined window of bases. The execution report only appears in the console. Using the input above with the default values, an output example of this is the following:

```

Minimum QS      : 33
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBII
Total reads     : 2
Trimmed reads   : 1

```

### 3.20 Program `gto2_fq_minimum_local_quality_score_reverse`

The `gto2_fq_minimum_local_quality_score_reverse` filters the reverse reads, considering the quality score average of a defined window size of bases.

For help type:

```
./gto2_fq_minimum_local_quality_score_reverse -h
```

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

#### Input parameters

The `gto2_fq_minimum_local_quality_score_reverse` 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_local_quality_score_reverse [options] [--] args]
      or: ./gto2_fq_minimum_local_quality_score_reverse [options]
```

It filters the reverse reads, considering the quality score



[illegible]

### 3.21 Program gto2\_fq\_xs

For help type:

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

### Input parameters

The **gto2\_fq\_xs** program needs a FASTQ file to compute.

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCGTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

### Output

The output of the **gto2\_fq\_xs** program is a FASTQ file. Using the input above using the common usage with 5 reads (-n 5), an output example of this is the following:

```


```

## 3.22 Program *gto2\_fq\_complement*

The **gto2\_fq\_complement** replaces the ACGT bases with their complements in a FASTQ file format.

For help type:

```
./gto2_fq_complement -h
```

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

## Input parameters

The **gto2\_fq\_complement** 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_complement [options] [--] args]
       or: ./gto2_fq_complement [options]
```

It replaces the ACGT bases with their complements in a FASTQ file format.

```
-h, --help          Show this help message and exit
```

### Basic options

```
< input.fastq      Input FASTQ file (stdin)
> output.fastq     Output FASTQ file (stdout)
```

```
Example: ./gto2_fq_complement < 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCGTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTACAGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## Output

The output of the **gto2\_fq\_complement** program is the FASTQ file with the ACGT base complements. 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
CCCACTACCGGCGACGGCTACCGCAGTTTAGGGTGGTTCAATGGGAATTGTTGAATTCCC
```



```

+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
CAAGTCCCTATGCTGCAACATAAAATTCTTAGACTTCGTCTTCAGCTACTATTATGCGC
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI

```

---

### 3.23 Program `gto2_fq_reverse`

The `gto2_fq_reverse` reverses the ACGT bases order for each read in a FASTQ file format.

For help type:

```
./gto2_fq_reverse -h
```

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

#### Input parameters

The `gto2_fq_reverse` 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_reverse [options] [--] args]
       or: ./gto2_fq_reverse [options]
```

It reverses the ACGT bases order for each read in a FASTQ file.

<code>-h, --help</code>	Show this help message and exit
-------------------------	---------------------------------

#### Basic options

<code>&lt; input.fastq</code>	Input FASTQ file (stdin)
-------------------------------	--------------------------

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

```
Example: ./gto2_fq_reverse < input.fastq > output.fastq
```

An example of such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCGTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI
```

## Output

The output of the **gto2\_fq\_reverse** program is the FASTQ file complement with the flag “(Reversed)” added in the header. Using the input above, an output example of this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 (Reversed)
GGGAATTCAACAATTCCCATGAACCACCCTAAACTGCGGTAGCCGTCGCCGTAGTGGG
+
IIIDIIIIIIIIIIIIIIIIIIIIICI9GI9IIIIIIIIIIIIIIIIIIIIIIIIIIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 (Reversed)
GCGCATAATAGTAGCTGAAGACGAAGTCTAAGAATTTATGTTTGCAGCATAGGGACTTG
+
IGIIIIIIIIIIIIIIIIIIIIIBI6IIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

---

## 3.24 Program gto2\_fq\_variation\_map

The **gto2\_fq\_variation\_map** identifies the variation that occurs in the sequences relative to the reads or a set of reads.

For help type:

```
./gto2_fq_variation_map -h
```

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

### Input parameters

The **gto2\_fq\_variation\_map** program needs FASTQ, FASTA or SEQ files to be used as reference and target files.

The attribution is given according to:

```
Usage: ./gto2_fq_variation_map <OPTIONS>... [FILE]:<...>
[FILE]:<...>
```

The **gto2\_fq\_variation\_map** is a tool to map relative singularity regions. The (probabilistic) Bloom filter is automatically set.

```
-v                verbose mode,
-a                about CHESTER,
-s <value>       bloom size,
-i                use inversions,
-p                show positions/words,
-k <value>       k-mer size (up to 30),

[rFile1]:<rFile2>:<...> reference file(s),
[tFile1]:<tFile2>:<...> target file(s).
```

The reference files may be FASTA, FASTQ or DNA SEQ, while the target files may be FASTA OR DNA SEQ. Report bugs to <{pratas,raquelsilva,ap,pjf}@ua.pt>.

An example of a reference file (Multi-FASTA format) is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCT
CCCTGCTGCCATTGTCCCCGGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCT
TGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAAGTGTTTGTGAGTGACCTCCGG
GCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCCG
```

An example for the target file (FASTQ format) is:

## Output

```
1111111111111111111111111000000000000000000000000000000000000000
1111111111111111111111111100000000000000000000000000000000000000
000000000000
```

The `gto2_fq_variation_filter` filters and segments the regions of singularity from the output of `gto2_fq_variation_map`.

```
./gto2_fq_variation_filter -h
```

### Input parameters

The attribution is given according to:

```
-v          verbose mode,
-a          about CHESTER,
-t <value> threshold [0.0;1.0],
-w <value> window size,
-u <value> sub-sampling,
```

The target files may be generated by `gto2fq_variation_map`.  
Report bugs to <{pratas,raquelsilva,ap,pjf}@ua.pt>.

```
111111111111111111111111111110000000000000000000000000000000000000
111111111111111111111111111110000000000000000000000000000000000000
000000000000
```

## Output

The output of the **gto2\_fq\_variation\_filter** program is a text file with the coordinates of the segmented regions. Using the inputs above, an output example of this is the following:

```
#132#132
30:60
90:130
```

---

### 3.26 Program gto2\_fq\_variation\_visual

The **gto2\_fq\_variation\_visual** depicts the regions of singularity using the output from **gto2\_fq\_variation\_filter** into an SVG image.

For help type:

```
./gto2_fq_variation_visual -h
```

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

#### Input parameters

The **gto2\_fq\_variation\_visual** program needs a the output of **gto2\_fq\_variation\_filter** to compute.

The attribution is given according to:

```
Usage: ./gto2_fq_variation_visual <OPTIONS>... [FILE]:<...>
The gto2_fq_variation_visual is a tool to visualize relative
singularity regions.
```

```
-v                verbose mode,
-a                about CHESTER,
-e <value>        enlarge painted regions,
```

```
[tFile1]:<tFile2>:<...> target file(s).
```

Report bugs to <{pratas,raquelsilva,ap,pjf}@ua.pt>.

An example of such an input file is:

```
#132#132
30:60
90:130
```

## Output

The output of the **gto2\_fq\_variation\_visual** program is a SVG plot with the maps. In the following Figure, is represented the plot using the input above.



**FIGURE 3.1:** Execution plot of the variation visual tool using the previous input.

## 3.27 Program *gto2\_fq\_metagenomics*

The **gto2\_fq\_metagenomics** is an ultra-fast method to infer metagenomic composition of sequenced reads relative to a database. *gto2\_fq\_metagenomics* measures similarity between any FASTQ file (or FASTA), independently from the size, against any multi-FASTA database, such as the entire set of complete genomes from the NCBI. *gto2\_fq\_metagenomics* supports single reads, paired-end reads, and compositions of both. It has been tested in many platforms, such as Illumina MySeq, HiSeq, Novaseq, IonTorrent.

*gto2\_fq\_metagenomics* is efficient to detect the presence and authenticate a given species in the FASTQ reads. The core of the method is

based on relative data compression. `gto2_fq_metagenomics` uses variable multi-threading, without multiplying the memory for each thread, being able to run efficiently in a common laptop.

For help type:

```
./gto2_fq_metagenomics -h
```

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

### Input parameters

The `gto2_fq_metagenomics` program needs a FASTQ file to compute.

The attribution is given according to:

#### NAME

The `gto2_fq_metagenomics` is a tool to infer metagenomic composition.

#### SYNOPSIS

```
gto2_fq_metagenomics [OPTION]... [FILE1]:[FILE2]:...  
[FILE]
```

#### SAMPLE

```
gto2_fq_metagenomics -v -F -l 47 -Z -y pro.com  
reads1.fq:reads2.fq DB.fa
```

#### DESCRIPTION

It infers metagenomic sample composition of sequenced reads. The core of the method uses a cooperation between multiple context and tolerant context models with several depths. The reference sequences must be in a multi-FASTA format. The sequenced reads must be trimmed and in FASTQ format.

Non-mandatory arguments:

`-h`                      give this help,



```

-F          force mode (overwrites top file),
-V          display version number,
-v          verbose mode (more information),
-Z          database local similarity,
-s          show compression levels,

-l <level>  compression level [1;47],
-p <sample> subsampling (default: 1),
-t <top>    top of similarity (default: 20),
-n <nThreads> number of threads (default: 2),

-x <FILE>   similarity top filename,
-y <FILE>   profile filename (-Z must be on).

Mandatory arguments:

[FILE1]:[FILE2]:... metagenomic filename (FASTQ),
                    Use ":" for splitting files.

[FILE]             database filename (Multi-FASTA).

```

An example of such an input file is:

```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=60
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGI

```

## Output

The output of the **gto2\_fq\_metagenomics** program is a CSV file (top.csv) with the highest probability of being contained in the samples. An example for this CSV file is the following:

```

2 66725 12.263 NC_037703.1_Saccharomyces_ludwigii...
1 66725 12.263 NC_037703.1_Saccharomyces_ludwigii...

```

3	107123	11.492	NC_012621.1_Nakaseomyces_bacillispor...
4	107123	11.492	NC_012621.1_Nakaseomyces_bacillispor...
5	16592	11.153	NC_024030.1_Equus_przewalskii_mitoch...
6	14583	10.851	NC_021120.1_Bursaphelenchus_mucronat...
7	162504	10.607	NC_018415.1_Candidatus_Carsonella_ru...
8	10315	10.586	NC_016117.1_Mnemiopsis_leidyi_mitoch...
9	162589	10.550	NC_018414.1_Candidatus_Carsonella_ru...
10	166163	10.476	NC_018416.1_Candidatus_Carsonella_ru...

# 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](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.