



The Genomics Toolkit

J. Almeida^{1,2} (j.r.dealmeida@udc.es)

D. Pratas¹ (pratas@ua.pt)

¹ IEETA, University of Aveiro, Portugal

² DICT, University of A Coruña, Spain

Version 1.1

Contents

1	Introduction	4
1.1	Installation	4
1.2	License	5
2	FASTQ tools	6
2.1	Program gto_fastq_to_fasta	7
2.2	Program gto_fastq_to_mfasta	8
2.3	Program gto_fastq_exclude_n	9
2.4	Program gto_fastq_extract_quality_scores	10
2.5	Program gto_fastq_info	11
2.6	Program gto_fastq_maximum_read_size	13
2.7	Program gto_fastq_minimum_quality_score	14
2.8	Program gto_fastq_minimum_read_size	15
2.9	Program gto_fastq_rand_extra_chars	16
2.10	Program gto_fastq_from_seq	17
2.11	Program gto_fastq_mutate	19
2.12	Program gto_fastq_split	20
2.13	Program gto_fastq_pack	22
2.14	Program gto_fastq_unpack	23
2.15	Program gto_fastq_quality_score_info	24
2.16	Program gto_fastq_quality_score_max	25
2.17	Program gto_fastq_quality_score_min	26
2.18	Program gto_fastq_cut	28
2.19	Program gto_fastq_minimum_local_quality_score_forward	29
2.20	Program gto_fastq_minimum_local_quality_score_reverse	30
2.21	Program gto_fastq_clust_reads	32
3	FASTA tools	34
3.1	Program gto_fasta_to_seq	35
3.2	Program gto_fasta_from_seq	36
3.3	Program gto_fasta_extract	37

3.4	Program gto_fasta_extract_by_read	38
3.5	Program gto_fasta_info	39
3.6	Program gto_fasta_mutate	41
3.7	Program gto_fasta_rand_extra_chars	42
3.8	Program gto_fasta_extract_read_by_pattern	43
3.9	Program gto_fasta_find_n_pos	45
3.10	Program gto_fasta_split_reads	46
3.11	Program gto_fasta_rename_human_headers	47
3.12	Program gto_fasta_extract_pattern_coords	48
4	Genomic sequence tools	50
4.1	Program gto_genomic_gen_random_dna	50
4.2	Program gto_genomic_rand_seq_extra_chars	51
4.3	Program gto_genomic_dna_mutate	52
4.4	Program gto_genomic_extract	54
4.5	Program gto_genomic_period	55
4.6	Program gto_genomic_count_bases	57
5	Amino acid sequence tools	59
5.1	Program gto_amino_acid_to_group	59
5.2	Program gto_amino_acid_to_pseudo_dna	60
6	General purpose tools	63
6.1	Program gto_char_to_line	64
6.2	Program gto_reverse	65
6.3	Program gto_new_line_on_new_x	66
6.4	Program gto_upper_bound	68
6.5	Program gto_lower_bound	69
6.6	Program gto_brute_force_string	70
6.7	Program gto_real_to_binary_with_threshold	71
6.8	Program gto_sum	72
6.9	Program gto_filter	73
6.10	Program gto_word_search	74
6.11	Program gto_permute_by_blocks	76
6.12	Program gto_info	77
6.13	Program gto_segment	78
6.14	Program gto_comparative_map	79
6.15	Program gto_xs	81
6.16	Program gto_geco	82
6.17	Program gto_gede	82

Chapter 1

Introduction

Recent advances in DNA sequencing have revolutionized the field of genomics, making it possible for research groups to generate large amounts of sequenced data, very rapidly and at substantially lower cost [?]. The storage of genomic data is being addressed using specific file formats, such as FASTQ and FASTA. Therefore, its analysis and manipulation is crucial [?]. Many frameworks for analysis and manipulation emerged, namely GALAXY [?], GATK [?], HTSeq [?], MEGA [?], among others. Several of these frameworks require licenses, while others do not provide a low level access to the information, since they are commonly approached by scripting or programming languages not efficient for the purpose. Moreover, several lack on variety, namely the ability to perform multiple tasks using only one toolkit.

We describe **GTO**, a complete toolkit for genomics, namely for FASTA-FASTQ formats and sequences (DNA, amino acids, text), with many complementary tools. The toolkit is for Linux- and Unix-based systems, built for ultra-fast computations. **GTO** supports pipes for easy integration with the sub-programs belonging to **GTO** as well as external tools. **GTO** works as the *LEGOs*, since it allows the construction of multiple pipelines with many combinations.

GTO includes tools for information display, randomization, edition, conversion, extraction, search, calculation, and visualization. **GTO** is prepared to deal with very large datasets, typically in the scale Gigabytes or Terabytes (but not limited).

The complete toolkit is an optimized command line version, using the prefix “gto_” followed by the suffix with the respective name of the program. **GTO** is implemented in C language and it is available, under the MIT license, at:

```
http://bioinformatics.ua.pt/gto
```

1.1 Installation

For **GTO** installation, run:

```
git clone https://github.com/bioinformatics-ua/gto.git
cd gto/src/
```

1.2 License

The license is **MIT**. In resume, it is a short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions:

- commercial use;
- modification;
- distribution;
- private use.

Limitations:

- liability;
- warranty.

Conditions:

- License and copyright notice.

For details on the license, consult: <https://opensource.org/licenses/MIT>.

Chapter 2

FASTQ tools

Current available tools for FASTQ format analysis and manipulation include:

1. `gto_fastq_to_fasta`: it converts a FASTQ file format to a pseudo FASTA file.
2. `gto_fastq_to_mfasta`: it converts a FASTQ file format to a pseudo Multi-FASTA file.
3. `gto_fastq_exclude_n`: it discards the FASTQ reads with the minimum number of "N" symbols.
4. `gto_fastq_extract_quality_scores`: it extracts all the quality-scores from FASTQ reads.
5. `gto_fastq_info`: it analyses the basic information of FASTQ file format.
6. `gto_fastq_maximum_read_size`: it filters the FASTQ reads with the length higher than the value defined.
7. `gto_fastq_minimum_quality_score`: it discards reads with average quality-score below of the defined.
8. `gto_fastq_minimum_read_size`: it filters the FASTQ reads with the length smaller than the value defined.
9. `gto_fastq_rand_extra_chars`: it substitutes in the FASTQ files, the DNA sequence the outside ACGT chars by random ACGT symbols.
10. `gto_fastq_from_seq`: it converts a genomic sequence to pseudo FASTQ file format.
11. `gto_fastq_mutate`: it creates a synthetic mutation of a FASTQ file given specific rates of mutations, deletions and additions.
12. `gto_fastq_split`: it splits Paired End files according to the direction of the strand ('/1' or '/2').
13. `gto_fastq_pack`: it packages each FASTQ read in a single line.
14. `gto_fastq_unpack`: it unpacks the FASTQ reads packaged using the `gto_fastq_pack` tool.

15. `gto_fastq_quality_score_info`: it analyses the quality-scores of a FASTQ file.
16. `gto_fastq_quality_score_min`: it analyses the minimal quality-scores of a FASTQ file.
17. `gto_fastq_quality_score_max`: it analyses the maximal quality-scores of a FASTQ file.
18. `gto_fastq_cut`: it cuts read sequences in a FASTQ file.
19. `gto_fastq_minimum_local_quality_score_forward`: it filters the reads considering the quality score average of a defined window size of bases.
20. `gto_fastq_minimum_local_quality_score_reverse`: it filters the reverse reads, considering the average window size score defined by the bases.
21. `gto_fastq_clust_reads`: it agroups reads and creates an index file.

2.1 Program `gto_fastq_to_fasta`

The `gto_fastq_to_fasta` converts a FASTQ file format to a pseudo FASTA file. However, it does not align the sequence. Also, it extracts the sequence and adds a pseudo header.

For help type:

```
./gto_fastq_to_fasta -h
```

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

Input parameters

The `gto_fastq_to_fasta` 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: ./gto_fastq_to_fasta [options] [--] args]
or: ./gto_fastq_to_fasta [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: ./gto_fastq_to_fasta < input.fastq > output.fasta
```

An example on such an input file is:


```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

Output

The output of the `gto_fastq_to_fasta` program a FASTA file.

Using the input above, an output example for this is the following:

```

GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT

```

2.2 Program `gto_fastq_to_mfasta`

The `gto_fastq_to_mfasta` onverts a FASTQ file format to a pseudo Multi-FASTA file. However, it does not align the sequence. Also, it extracts the sequence and adds a pseudo header.

For help type:

```
./gto_fastq_to_mfasta -h
```

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

Input parameters

The `gto_fastq_to_mfasta` 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: ./gto_fastq_to_mfasta [options] [--] args]
      or: ./gto_fastq_to_mfasta [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: ./gto_fastq_to_mfasta < input.fastq > output.mfasta
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

Output

The output of the `gto_fastq_to_mfasta` program a Multi-FASTA file.

Using the input above, an output example for this is the following:

```
>SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCCTTAACAACCTTAAGGGTTTTCAAATAGA
>SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
```

2.3 Program `gto_fastq_exclude_n`

The `gto_fastq_exclude_n` discards the FASTQ reads with the minimum number of "N" symbols. Also, if present, it will erase the second header (after +).

For help type:

```
./gto_fastq_exclude_n -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_exclude_n [options] [--] args]
or: ./gto_fastq_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: ./gto_fastq_exclude_n < input.fastq > output.fastq
```

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

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCAACCAANATACCCCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

Output

The output of the `gto_fastq_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=72
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
Total reads      : 2
Filtered reads   : 1
```

2.4 Program `gto_fastq_extract_quality_scores`

The `gto_fastq_extract_quality_scores` extracts all the quality-scores from FASTQ reads.

For help type:

```
./gto_fastq_extract_quality_scores -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_extract_quality_scores [options] [--] args]
      or: ./gto_fastq_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: ./gto_fastq_extract_quality_scores < input.fastq > output.fastq

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

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-1)8I
```

Output

The output of the `gto_fastq_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 for this is the following:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-1)8I
Total reads          : 2
Total Quality-Scores : 144
```

2.5 Program `gto_fastq_info`

The `gto_fastq_info` analyses the basic information of FASTQ file format.

For help type:

```
./gto_fastq_info -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_info [options] [--] args]
      or: ./gto_fastq_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: ./gto_fastq_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 on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTACAGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

Output

The output of the `gto_fastq_info` program is a set of information related to the file read.

Using the input above, an output example for 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
```

2.6 Program gto fastq maximum read size

The `gto_fastq_maximum_read_size` filters the FASTQ reads with the length higher than the value defined. For help type:

```
./gto_fastq_maximum_read_size -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_maximum_read_size [options] [--] args]
      or: ./gto_fastq_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: `./gto_fastq_maximum_read_size -s <size> < input.fastq > output.fastq`

Console output example :

<FASTQ non-filtered reads>

Total reads : value

Filtered reads	: value
----------------	---------

An example on such an input file is:

[illegible]

Output

The output of the `gto_fastq_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 60, an output example for this is the following:

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

2.7 Program gto_fastq_minimum_quality_score

The `gto_fastq_minimum_quality_score` discards reads with average quality-score below of the defined. For help type:

```
./gto_fastq_minimum_quality_score -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_minimum_quality_score [options] [--] args]
      or: ./gto_fastq_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: ./gto_fastq_minimum_quality_score -m <min> < input.fastq > output.fastq

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

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: `./gto_fastq_minimum_quality_score -m <min> < input.fastq > output.fastq`

Console output example:

```
<FASTQ non-filtered reads>
```

```
Total reads      : value
```

```
Filtered reads : value
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAAATCTGAAGCAGAAATCGATGATAATACGCGTCGTTTTATCAT
```

```
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
54599<>77977==6=?I6IBI::33344235521677999>>><<@@A@BBCDGGBFFH>IIIII-I)8I
```

Output

The output of the `gto_fastq_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 average value as 30, an output example for this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
Total reads      : 2
Filtered reads   : 1
```

2.8 Program `gto_fastq_minimum_read_size`

The `gto_fastq_minimum_read_size` filters the FASTQ reads with the length smaller than the value defined. For help type:

```
./gto_fastq_minimum_read_size -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_minimum_read_size [options] [--] args]
or: ./gto_fastq_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: ./gto_fastq_minimum_read_size -s <size> < input.fastq > output.fastq

Console output example:
```



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

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

Output

The output of the `gto_fastq_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 65, an output example for this is the following:

```
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
Total reads      : 2
Filtered reads   : 1
```

2.9 Program `gto_fastq_rand_extra_chars`

The `gto_fastq_rand_extra_chars` substitutes in the FASTQ files, the DNA sequence the outside ACGT chars by random ACGT symbols.

For help type:

```
./gto_fastq_rand_extra_chars -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_rand_extra_chars [options] [--] args]
or: ./gto_fastq_rand_extra_chars [options]
```

It substitutes 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: ./gto_fastq_rand_extra_chars < input.fastq > output.fastq

```

An example on such an input file is:

```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

Output

The output of the `gto_fastq_rand_extra_chars` program is a FASTQ file.

Using the input above, an output example for this is the following:

```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GTGTGATGGCCGCTGCCGATGGCGCATAATCCCACCAACATACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGATTGTATTTTAAAGAATCTGCAGCAGAAGTCGATGATAATACGCGCCGTTTATCAG
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

2.10 Program `gto_fastq_from_seq`

The `gto_fastq_from_seq` converts a genomic sequence to pseudo FASTQ file format.

For help type:

```

./gto_fastq_from_seq -h

```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_from_seq [options] [--] args]
      or: ./gto_fastq_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: ./gto_fastq_from_seq -l <lineSize> -n <name> < input.seq > output.fastq

```

An example on such an input file is:

```

ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA

```

Output

The output of the `gto_fastq_from_seq` program is a pseudo FASTQ file.

An example, using the size line as 80 and the read's header as "SeqToFastq", for the input, is:

```

@SeqToFastq1
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq2
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq3
GTGGTTTGAGTGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq4
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq5

```

```

TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq6
CTGCTGCTCTCCGGGGCCACGGCCACCCTGCCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq7
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq8
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq9
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq10
TAATTACAGACCTGAA
+
FFFFFFFFFFFFFFFFFFFF

```

2.11 Program gto_fastq_mutate

The `gto_fastq_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 they are optional.

For help type:

```
./gto_fastq_mutate -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_mutate [options] [--] args]
      or: ./gto_fastq_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, --seed=<int>       Starting point to the random generator
  -m, --mutation-rate=<dbl> Defines the mutation rate (default 0.0)
  -d, --deletion-rate=<dbl> Defines the deletion rate (default 0.0)
  -i, --insertion-rate=<dbl> Defines the insertion rate (default 0.0)
  -a, --ACGTN-alphabet   When active, the application uses the ACGTN alphabet

Example: ./gto_fastq_mutate -s <seed> -m <mutation rate> -d <deletion rate> -i
<insertion rate> -a < input.fastq > output.fastq

```

An example on such an input file is:

```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGCTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

Output

The output of the `gto_fastq_mutate` program is a FASTQ file with the synthetic mutation of input file. Using the input above with the seed value as 1 and the mutation rate as 0.5, an output example for this is the following:

```

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGACTTTGAGGTGTGGCGATAGACTGAAACACTTCAGGGTAAAATCACTCGCAAAAGTGCTATGGTTATGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGAGCCTTTACCGTAGGGGTGTAAGATTTTATACAAAAAGTCCAGGTCAAGAGGAATCGGACAACCGA
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

2.12 Program `gto_fastq_split`

The `gto_fastq_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:

```
./gto_fastq_split -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_split [options] [--] args]
       or: ./gto_fastq_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: ./gto_fastq_split -t <output_forward.fastq> -r <output_reverse.fastq> < input.fastq > output

Output example :
Total reads      : value
Singleton reads  : value
Forward reads    : value
Reverse reads    : value
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72 1
GNNTGATGGCCGCTGCCGATGGCGNANAAATCCCACCAANATACCCCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-1)8I
```

Output

The output of the `gto_fastq_split` program is a set of information related to the file read.

Using the input above, an output example for this is the following:

```
Total reads      : 2
Singleton reads   : 0
Forward reads     : 65536
Reverse reads     : 1
```

Also, this program generates two FASTQ files, with the reverse and forward reads.

An example of the forward reads, for the input, is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72 1
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
```

2.13 Program gto_fastq_pack

The `gto_fastq_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:

```
./gto_fastq_pack -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_pack [options] [--] args]
or: ./gto_fastq_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.fastq           Output FASTQ file format (stdout)

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

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

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAAGATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-1)8I
```

Output

The output of the `gto_fastq_pack` program is a packaged FASTQ file.

Using the input above, an output example for this is the following:

```
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72+ 0
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-I)8I
SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72+ 1
```

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

```
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72+ 0
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-I)8I
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72+ 1
```

2.14 Program `gto_fastq_unpack`

The `gto_fastq_unpack` unpacks the FASTQ reads packaged using the `gto_fastq_pack` tool.

For help type:

```
./gto_fastq_unpack -h
```

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

Input parameters

The `gto_fastq_unpack` 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: ./gto_fastq_unpack [options] [--] args
or: ./gto_fastq_unpack [options]
```

It unpacks the FASTQ reads packaged using the `gto_fastq_pack` tool.

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

Basic options

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

Optional

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



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

An example on such an input file is:

```
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTAAGGGTTNTCAAATAGA
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72+ 0
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72+ 1
```

Output

The output of the `gto_fastq_unpack` program is a packaged FASTQ file.

Using the input above, an output example for this is the following:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTAAGGGTTNTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

2.15 Program `gto_fastq_quality_score_info`

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

For help type:

```
./gto_fastq_quality_score_info -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_quality_score_info [options] [--] args]
or: ./gto_fastq_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: ./gto_fastq_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 on such an input file is:

```

@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=72 1
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
+111
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=72 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+222
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I

```

Output

The output of the `gto_fastq_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 for 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
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
--+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73

```

2.16 Program `gto_fastq_quality_score_max`

The `gto_fastq_quality_score_max` analyses the maximal quality-scores of a FASTQ file. For help type:

```
./gto_fastq_quality_score_max -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_quality_score_max [options] [--] args]
       or: ./gto_fastq_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: ./gto_fastq_quality_score_max -m <max> < input.fastq > output
```

An example on such an input file is:

```
@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=72 1
GNNTGATGGCGCGTGC CGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIII9IG9ICIIIIIIIIIIABAAABCIIIIFFGIIAACBBIIII6IBIIIIII>IIIIII/
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=72 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTATCAN
+
IIIIIIIIABAAABCIIIIFFGIIAACBBIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

Output

The output of the `gto_fastq_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 30, an output example for this is the following:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
--+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
73 73 73 73 73 73 73 73 73 73 73 73 66 73 73 73 73 73 73 73 73 73 73 73 73 73 73 73
```

2.17 Program `gto_fastq_quality_score_min`

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

For help type:

```
./gto_fastq_quality_score_min -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_quality_score_min [options] [--] args]
       or: ./gto_fastq_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: ./gto_fastq_quality_score_min -m <max> < input.fastq > output
```

An example on such an input file is:

```
@111 071112_SLXA-EAS1_s_7:5:1:817:345 length=72 1
GNNTGATGGCGCGCTGCCGATGGCGNANAATCCCACCAANATACCCCTTAACAACCTTAAGGGTTNTCAAATAGA
+
IIIIIIIIII9IG9ICIIIIIIIIIIABAAABCIIIIFFGIIAACBBIIII6IBIIIIII>IIIIII/
@222 071112_SLXA-EAS1_s_7:5:1:801:338 length=72 2
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+
IIIIIIIIABAAABCIIIIFFGIIAACBBIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

Output

The output of the `gto_fastq_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 30, an output example for this is the following:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
--+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
73 73 73 73 73 73 73 65 66 65 65 65 57 67 71 57 73 67 70 70 71 73 73 65 65 67 66 66 73 65
```

2.18 Program gto_fastq_cut

The `gto_fastq_cut` cuts read sequences in a FASTQ file. It requires that the initial and end positions for the cut.

For help type:

```
./gto_fastq_cut -h
```

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

Input parameters

The `gto_fastq_cut` program needs 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: ./gto_fastq_cut [options] [--] args]
or: ./gto_fastq_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: ./gto_fastq_cut -i <initial> -e <end> < input.fastq > output.fastq
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAAGGATACGACGTTTGTATTTTAAAGATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-1)8I
```

Output

The output of the `gto_fastq_cut` program is a FASTQ file cut.

Using the initial value as 10 and the end value as 30, an example for this input, is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
CGCTGCCGATGGCGTCAAATC
+
IIIIIIIIIIIIIIIIIIII9
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
ACGACGTTTGTATTTTAAGAA
+
IIIIIIIIIIIIIIIIIIII
```

2.19 Program gto fastq minimum local quality score forward

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

For help type:

```
./gto_fastq_minimum_local_quality_score_forward -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_minimum_local_quality_score_forward [options] [--] args]
or: ./gto_fastq_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
    -k, --windowsize=<int>    The window size of bases (default 5)
    -w, --minavg=<int>        The minimum average of quality score (default 25)
    -m, --minqs=<int>         The minimum value of the quality score (default 33)
    < input.fastq              Input FASTQ file format (stdin)
    > output.fastq             Output FASTQ file format (stdout)

Example: ./gto_fastq_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
```

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

```
-k, --window-size=<int>      The window size of bases (default 5)
-w, --min-avg=<int>          The minimum average of quality score (default 25)
-m, --min-q=<int>            The minimum value of the quality score (default 33)
< input.fastq                Input FASTQ file format (stdin)
> output.fastq                Output FASTQ file format (stdout)
```

```
Example: ./gto_fastq_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 on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII-I)8I
```

Output

The output of the `gto_fastq_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 for this is the following:

```
Minimum QS      : 33
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCTTAACAACCTTAAGGGTTTTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTT
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIII
Total reads     : 2
Trimmed reads   : 1
```

2.20 Program `gto_fastq_minimum_local_quality_score_reverse`

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

For help type:

```
./gto_fastq_minimum_local_quality_score_reverse -h
```

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

Input parameters

The `gto_fastq_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: ./gto_fastq_minimum_local_quality_score_reverse [options] [--] args]
or: ./gto_fastq_minimum_local_quality_score_reverse [options]
```

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

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

Basic options

```
-k, --windowsize=<int>    The window size of bases (default 5)
-w, --minavg=<int>        The minimum average of quality score (default 25)
-m, --minqs=<int>         The minimum value of the quality score (default 33)
< input.fastq             Input FASTQ file format (stdin)
> output.fastq            Output FASTQ file format (stdout)
```

```
Example: ./gto_fastq_minimum_local_quality_score_reverse -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 on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAAGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

Output

The output of the `gto_fastq_minimum_local_quality_score_reverse` 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 for this is the following:

```
Minimum QS: 33
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
Total reads      : 2
Trimmed reads    : 1
```


2.21 Program gto fastq clust reads

The `gto_fastq_clust_reads` agroups reads and creates an index file. It cluster reads in terms of Seq k-mer Lexicographical order.

For help type:

```
./gto_fastq_clust_reads -h
```

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

Input parameters

The `gto_fastq_clust_reads` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTQ file. The program sorts the FASTQ reads according to the lexicographic order of the genomic sequences.

The attribution is given according to:

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

It agroups reads and creates an index file.
It cluster reads in therms of Seq k-mer Lexicographical order

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

Basic options
-c, --ctx=<int>
< input.fastq       Input FASTQ file format (stdin)
> output.fastq      Output FASTQ file format (stdout)

Example: ./gto_fastq_clust_reads -c <ctx> < input.fastq > output.fastq
```

An example on such an input file is:

```
@SRR001661.1 071112_SLXA-EAS1_s_7:5:1:817:345
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIII
@SRR001661.2 071112_SLXA-EAS1_s_7:5:1:801:338
GTTACAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIIGI
@SRR001661.3 071112_SLXA-EAS1_s_7:5:1:821:328
AACGCGTATTTCGGAGCTTCTTCGTTGGGTACGTGCGCCTATTATGCGGCGCGATTGCTAT
+
IIIIII6BBB6BBBBBBBBBBBBBBBBBDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
@SRR001661.4 071112_SLXA-EAS1_s_7:5:1:943:128
ATCGCGCATTCGACTGGTACGTGTACGTGTAGTCGTAGCGTATGTTTCGGTCGTATGCGTG
+
```

```
II77777LPMMPMMMMIIIIIIIIIIII77777777BBBBBBBDDDDDDIIIIII
```

Output

The output of the `gto_fastq_clust_reads` program is a FASTQ file with clustered reads in terms of the genomic sequence k-mer Lexicographical order. An example, for the output, is:

```
@SRR001661.3 071112_SLXA-EAS1_s_7:5:1:821:328
AACGCGTATTCGGAGCTTCTTCGTTGGGTACGTGCGCCTATTATGCGGCGCGATTGCTAT
+
IIIIIII6BBB6BBBBBBBBBBBBBBBBBDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
@SRR001661.4 071112_SLXA-EAS1_s_7:5:1:943:128
ATCGCGCATTGCGACTGGTACGTGTACGTGTAGTCGTAGCGTATGTTCCGGTCGTATGCGTG
+
II77777LPMMPMMMMIIIIIIIIIIII77777777BBBBBBBDDDDDDIIIIII
@SRR001661.1 071112_SLXA-EAS1_s_7:5:1:817:345
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001661.2 071112_SLXA-EAS1_s_7:5:1:801:338
GTTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIGI
```

Chapter 3

FASTA tools

Current available FASTA tools, for analysis and manipulation, are:

1. `gto_fasta_to_seq`: it converts a FASTA or Multi-FASTA file format to a seq.
2. `gto_fasta_from_seq`: it converts a genomic sequence to pseudo FASTA file format.
3. `gto_fasta_extract`: it extracts sequences from a FASTA file, which the range is defined by the user in the parameters.
4. `gto_fasta_extract_by_read`: it extracts sequences from each read in a Multi-FASTA file (splited by `\n`), which the range is defined by the user in the parameters.
5. `gto_fasta_info`: it shows the readed information of a FASTA or Multi-FASTA file format.
6. `gto_fasta_mutate`: it reates a synthetic mutation of a fasta file given specific rates of editions, deletions and additions.
7. `gto_fasta_rand_extra_chars`: it substitues in the DNA sequence the outside ACGT chars by random ACGT symbols.
8. `gto_fasta_extract_read_by_pattern`: it extracts reads from a Multi-FASTA file format given a pattern in the header.
9. `gto_fasta_find_n_pos`: it reports the "N" regions in a sequence or FASTA (seq) file.
10. `gto_fasta_split_reads`: it splits a Multi-FASTA file to multiple FASTA files.
11. `gto_fasta_rename_human_headers`: it changes the headers of FASTA or Multi-FASTA file to simple chrX by order, where X is the number.
12. `gto_fasta_extract_pattern_coords`: it extracts the header and coordinates from a Multi-FASTA file format given a pattern/motif in the sequence.

3.1 Program gto_fasta_to_seq

The `gto_fasta_to_seq` converts a FASTA or Multi-FASTA file format to a sequence.

For help type:

```
./gto_fasta_to_seq -h
```

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

Input parameters

The `gto_fasta_to_seq` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.

The attribution is given according to:

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

It converts a FASTA or Multi-FASTA file format to a seq.

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

Basic options
      < input.fasta             Input FASTA or Multi-FASTA file format (stdin)
      > output.seq              Output sequence file (stdout)

Example: ./gto_fasta_to_seq < input.fasta > output.seq
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCCCTGGAGGGT
GGCCCCACCGCGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGACCTCCCAGGCCAGTGCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_to_seq` program is a group sequence.

Using the input above, an output example for this is the following:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCCTCGGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGAACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

3.2 Program gto_fasta_from_seq

The `gto_fasta_from_seq` converts a genomic sequence to pseudo FASTA file format.

For help type:

```
./gto_fasta_from_seq -h
```

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

Input parameters

The `gto_fasta_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: ./gto_fasta_from_seq [options] [--] args]
or: ./gto_fasta_from_seq [options]

It converts a genomic sequence to pseudo FASTA file format.

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

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

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

Example: ./gto_fasta_from_seq -l <lineSize> -n <name> < input.seq > output.fasta
```

An example on such an input file is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
```

```
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_from_seq` program is a pseudo FASTA file.

Using the input above with the size line as 80 and the read's header as "SeqToFasta", an output example for this is the following:

```
>SeqToFasta
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA
```

3.3 Program `gto_fasta_extract`

The `gto_fasta_extract` extracts sequences from a FASTA file, which the range is defined by the user in the parameters.

For help type:

```
./gto_fasta_extract -h
```

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

Input parameters

The `gto_fasta_extract` program needs two parameters, which defines the begin and the end of the extraction, and two streams for the computation, namely the input and output standard. The input stream is a FASTA file.

The attribution is given according to:

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

It extracts sequences from a FASTA file.

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

```

Basic options
-i, --init=<int>      The first position to start the extraction (default 0)
-e, --end=<int>       The last extract position (default 100)
< input.fasta        Input FASTA or Multi-FASTA file format (stdin)
> output.seq         Output sequence file (stdout)

```

```

Example: ./gto_fasta_extract -i <init> -e <end> < input.fasta > output.seq

```

An example on such an input file is:

```

>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA

```

Output

The output of the `gto_fasta_extract` program is a group sequence.

Using the input above with the value 0 as the extraction starting point and the 50 as the ending, an output example for this is the following:

```

ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGG

```

3.4 Program `gto_fasta_extract_by_read`

The `gto_fasta_extract_by_read` extracts sequences from a FASTA or Multi-FASTA file, which the range is defined by the user in the parameters.

For help type:

```

./gto_fasta_extract_by_read -h

```

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

Input parameters

The `gto_fasta_extract_by_read` program needs two paramenters, which defines the begin and the end of the extraction, and two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.

The attribution is given according to:

```

Usage: ./gto_fasta_extract_by_read [options] [--] args]
or: ./gto_fasta_extract_by_read [options]

```

```

It extracts sequences from each read in a Multi-FASTA file (splited by \n)

```

```

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

Basic options
-i, --init=<int>    The first position to start the extraction (default 0)
-e, --end=<int>     The last extract position (default 100)
< input.fasta      Input FASTA or Multi-FASTA file format (stdin)
> output.fasta      Output FASTA or Multi-FASTA file format (stdout)

Example: ./gto_fasta_extract_by_read -i <init> -e <end> < input.fasta > output.fasta

```

An example on such an input file is:

```

>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCTCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA

```

Output

The output of the `gto_fasta_extract_by_read` program is FASTA or Multi-FASTA file with the extracted sequences.

Using the input above with the value 0 as the extraction starting point and the 50 as the ending, an output example for this is the following:

```

>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGG
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCC

```

3.5 Program `gto_fasta_info`

The `gto_fasta_info` shows the readed information of a FASTA or Multi-FASTA file format. For help type:

```
./gto_fasta_info -h
```

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

Input parameters

The `gto_fasta_info` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.

The attribution is given according to:

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

It shows read information of a FASTA or Multi-FASTA file format.

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

Basic options
    < input.fasta             Input FASTA or Multi-FASTA file format (stdin)
    > output                  Output read information (stdout)

Example: ./gto_fasta_info < input.fasta > output

Output example :
Number of reads      : value
Number of bases     : value
MIN of bases in read : value
MAX of bases in read : value
AVG of bases in read : value
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGACCTCCCAGGCCAGTGCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_info` program is a set of information related to the file read.

Using the input above, an output example for this is the following:

```
Number of reads      : 2
Number of bases     : 736
MIN of bases in read : 368
```

```
MAX of bases in read : 368
AVG of bases in read : 368.0000
```

3.6 Program `gto_fasta_mutate`

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

For help type:

```
./gto_fasta_mutate -h
```

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

Input parameters

The `gto_fasta_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 FASTA or Multi-FASTA File.

The attribution is given according to:

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

Creates a synthetic mutation of a fasta file given specific rates of editions,
deletions and additions

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

Basic options
      < input.fasta             Input FASTA or Multi-FASTA file format (stdin)
      > output.fasta            Output FASTA or Multi-FASTA file format (stdout)

Optional
      -s, --seed=<int>          Starting point to the random generator
      -e, --edit-rate=<dbl>     Defines the edition rate (default 0.0)
      -d, --deletion-rate=<dbl> Defines the deletion rate (default 0.0)
      -i, --insertion-rate=<dbl> Defines the insertion rate (default 0.0)
      -a, --ACGTN-alphabet      When active, the application uses the ACGTN alphabet

Example: ./gto_fasta_mutate -s <seed> -e <edit rate> -d <deletion rate> -i
<insertion rate> -a < input.fasta > output.fasta
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGTCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGCCAGGCGGCAGGAAGCAGGCCAGTGCC
```

```
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGTGCTGTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCCTGGAGGGT
GGCCCCACCGCGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTG
GTGGTTTGTAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_mutate` program is a FASTA or Multi-FASTA file with the synthetic mutation of input file.

Using the input above with the seed value as 1 and the edition rate as 0.5, an output example for this is the following:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACGCAACGNATTCTGCTGATCATANTGTNCCGCNCCCCNGCGACGGGGNCTCNCNNGCACACATNGTACCATTGTCCAC
NCTTNCANGTNANCGCTAGCAGGCTACNGTTTNTCCTCNCCTANNCCAANCNGGCGTNNNTACACTGGCACGTGCAGGCA
TNGGTGCGGCGNGNCCCTCCGGNAACGGCACCGGAGACGAAGCTCGGNGGNTATACAGGTGTCANGAAACATCCCCGCGNC
GNGTGNCNNGAANCCANAGAGTATCTCACTCACAACCCTGCGTGACNTCTAGAGNANGACCTTACNCACCNCTCCCNNT
NNGTACCACACCAATGAACGCTGCAGAAAGTCTGTTTNNAGGNGNGCA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ATTTGAAGGCAANCGGNCCAGNAATNCGGNGGGTGCGCTCNGTCTGTTGGCTACGGNCATCGGGCCCTGCTNTANTAAGCN
TGAAACCACCGNTCGNNGCACTTAGCAATNGCGNAANCCGTCGGCACGGCGGAGACNAANCCGCTANTNNTTTCCCGCTNA
ATGGNTGTACAAGACCNACTANACCANCCTCCGTCACCACACTGGAGCGCANGATGGNNCGCTGNCTAGNAGCNNTGAG
GCGCTCCNTCCTANAAANCCGTGGNCGAGCNCCCTATGGNAGNGTGGGGGTTTTACCGGAAGACCNTCGNGCCCTATGGG
AGCAATCANAANCTAGAAAGCTTACNGATGGTGANGAANTAGACTANG
```

3.7 Program `gto_fasta_rand_extra_chars`

The `gto_fasta_rand_extra_chars` substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols. It works both in FASTA and Multi-FASTA file formats.

For help type:

```
./gto_fasta_rand_extra_chars -h
```

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

Input parameters

The `gto_fasta_rand_extra_chars` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.

The attribution is given according to:

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

It substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols.
It works both in FASTA and Multi-FASTA file formats

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

Basic options

< input.fasta Input FASTA or Multi-FASTA file format (stdin)
> output.fasta Output FASTA or Multi-FASTA file format (stdout)

Example: ./gto_fasta_rand_extra_chars < input.fasta > output.fasta

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ANAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGAGGGTCCNCCGCTGCCCTGCTGCCATTGNCNCC
NGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGGCAGGAA
GNGGTTTGTAGTGGACCTCCNGGCCCTCATAGGAGAGGAAGCNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNC
GCGAATCCGNGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCN
TAAANNNTACCCATGAATGCTCACGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGNGCGCGGGACAGAATCTCCTTCTCCACCCCCCNNTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACC
NGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGG
ANAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGCNCAGGGTCCNCCGCTGCCCTGCTGCCATTGN
GAGGAAGCNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNCNGGTTTGAGTGGACCTCCNGGCCCTCATAGGA
TCACGCAANTTTAATTACAGACCTGAATAAANNNTACCCATGAATGC
```

Output

The output of the `gto_fasta_rand_extra_chars` program is a FASTA or Multi-FASTA file.

Using the input above, an output example for this is the following:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ATAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCCCGCTGCCCTGCTGCCATTGTCCCC
TGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGGCAGGAA
GAGGTTTGTAGTGGACCTCCCGGCCCTCATAGGAGAGGAAGCCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGTC
GCGAATCCGGGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCTTG
TAAAAGATCACCCATGAATGCTCACGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGTGCGCGGGACAGAATCTCCTTCTCCACCCCCCATCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACC
GGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGG
AGAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGTCCCTGGCTCCAGGGTCTCCGCTGCCCTGCTGCCATTGC
GAGGAAGCGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCGCGGTTTGAGTGGACCTCCTGGCCCTCATAGGA
TCACGCAACTTTAATTACAGACCTGAATAAATGTCACCCATGAATGC
```

3.8 Program `gto_fasta_extract_read_by_pattern`

The `gto_fasta_extract_read_by_pattern` extracts reads from a Multi-FASTA file format given a pattern in the header. Also, this pattern is case insensitive.

For help type:

```
./gto_fasta_extract_read_by_pattern -h
```

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

Input parameters

The `gto_fasta_extract_read_by_pattern` program needs two streams for the computation, namely the input and output standard. The input stream is a Multi-FASTA file.

The attribution is given according to:

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

It extracts reads from a Multi-FASTA file format given a pattern in the header (ID).

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

Basic options
-p, --pattern=<str>       Pattern to search in the file header
< input.fasta             Input Multi-FASTA file format (stdin)
> output.fasta            Output Multi-FASTA file format (stdout)

Example: ./gto_fasta_extract_read_by_pattern -p <pattern> < input.fasta > output.fasta
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_extract_read_by_pattern` program is a Multi-FASTA file.

Using the input above with the pattern value as "264", an output example for this is the following:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

3.9 Program gto_fasta_find_n_pos

The `gto_fasta_find_n_pos` reports the "N" regions in a sequence or FASTA (seq) file.

For help type:

```
./gto_fasta_find_n_pos -h
```

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

Input parameters

The `gto_fasta_find_n_pos` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTA file or a sequence.

The attribution is given according to:

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

It reports the 'N' regions in a sequence or FASTA (seq) file.

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

Basic options
      < input.fasta             Input FASTQ file format or a sequence (stdin)
      > output                  Output report of 'N' positions (stdout)

Example: ./gto_fasta_find_n_pos < input.fasta > output

The output obeys the following structure:
Begin      End Positions
<value> <value> <value>
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
NCNNNACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTNGTTTGAGTGGACCTCCGGGCCCCCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACNTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAN
```

Output

The output of the `gto_fasta_find_n_pos` program is a structured report of "N" appearances in the sequence or FASTA file. The first column is the first position of the "N" appearance, the second is the position of the last "N" in the interval found, and the last column is the count of "N" in this interval.

Using the input above, an output example for this is the following:

```

1    1    1
3    5    3
82   82   1
163  163  1
289  289  1

```

3.10 Program gto_fasta_split_reads

The `gto_fasta_split_reads` splits a Multi-FASTA file to multiple FASTA files.

For help type:

```
./gto_fasta_split_reads -h
```

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

Input parameters

The `gto_fasta_split_reads` program needs one stream for the computation, namely the input standard. This input stream is a Multi-FASTA file.

The attribution is given according to:

```

Usage: ./gto_fasta_split_reads [options] [--] args]
       or: ./gto_fasta_split_reads [options]

It splits a Multi-FASTA file to multiple FASTA files.

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

Basic options
    < input.fasta             Input Multi-FASTA file format (stdin)

Optional options
    -l, --location=<str>     Location to store the files

Example: ./gto_fasta_split_reads < input.fasta

```

An example on such an input file is:

```

>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAA

```

```
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_split_reads` program is a report summary of the execution, and the files created in the defined location.

Using the input above, an output example for this is the following:

```
1 : Splitting to file:./out1.fasta
2 : Splitting to file:./out2.fasta
```

3.11 Program `gto_fasta_rename_human_headers`

The `gto_fasta_rename_human_headers` changes the headers of FASTA or Multi-FASTA file to simple chrX by order, where X is the number.

For help type:

```
./gto_fasta_rename_human_headers -h
```

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

Input parameters

The `gto_fasta_rename_human_headers` program needs two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.

The attribution is given according to:

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

It changes the headers of FASTA or Multi-FASTA file to simple chr$1 by order.

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

Basic options
< input.fasta       Input FASTA or Multi-FASTA file format (stdin)
> output.fasta       Output FASTA or Multi-FASTA file format (stdout)

Example: ./gto_fasta_rename_human_headers < input.fasta > output.fasta
```

An example on such an input file is:

```
> AB000264 | acc = AB000264 | descr = Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGTCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```



```
> AB000263 | acc = AB000263 | descr = Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `gto_fasta_rename_human_headers` program is a FASTA or Multi-FASTA file. Using the input above, an output example for this is the following:

```
>chr1
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>chr2
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

3.12 Program `gto_fasta_extract_pattern_coords`

The `gto_fasta_extract_pattern_coords` extracts the header and coordinates from a Multi-FASTA file format given a pattern/motif in the sequence.

For help type:

```
./gto_fasta_extract_pattern_coords -h
```

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

Input parameters

The `gto_fasta_extract_pattern_coords` program needs two streams for the computation, namely the input and output standard. The input stream is a Multi-FASTA file.

The attribution is given according to:

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

It extracts the header and coordinates from a Multi-FASTA file format given a pattern/motif in the sequence.

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

Basic options

```
-p, --pattern=<str>  Pattern to search in the file header  
< input.fasta        Input Multi-FASTA file format (stdin)  
> output.coords      Output coordinates (stdout)
```

```
Example: ./gto_fasta_extract_pattern_coords -p <pattern> < input.fasta > output.coords
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA  
ACAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC  
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA  
GTGGTTTGAGTGGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC  
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC  
TAAACCTCACCCATGAATGCTCGCAACACGCAAGTTTAATTCGCAAGTTAGACCTGAACGGGAGGTGGCCACGCAAGTT
```

Output

The output of the `gto_fasta_extract_pattern_coords` program is a Multi-FASTA file.

Using the input above, an output example for this is the following:

```
177 183 >AB000264 |acc=AB000264|descr=Homo sapiens mRNA
```

Chapter 4

Genomic sequence tools

Current available genomic sequence tools, for analysis and manipulation, are:

1. `gto_genomic_gen_random_dna`: it generates a synthetic DNA.
2. `gto_genomic_rand_seq_extra_chars`: it substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols.
3. `gto_genomic_dna_mutate`: it creates a synthetic mutation of a sequence file given specific rates of mutations, deletions and additions.
4. `gto_genomic_extract`: it extracts sequences from a sequence file, which the range is defined by the user in the parameters.
5. `gto_genomic_period`: it calculates the best order depth of a sequence, using FCMs.
6. `gto_genomic_count_bases`: it counts the number of bases in sequence, FASTA or FASTQ files.

4.1 Program `gto_genomic_gen_random_dna`

The `gto_genomic_gen_random_dna` generates a synthetic DNA.

For help type:

```
./gto_genomic_gen_random_dna -h
```

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

Input parameters

The `gto_genomic_gen_random_dna` program needs one stream for the computation, namely the output standard.

The attribution is given according to:

```

Usage: ./gto_genomic_gen_random_dna [options] [--] args]
      or: ./gto_genomic_gen_random_dna [options]

It generates a synthetic DNA.

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

Basic options
      > output.seq              Output synthetic DNA sequence (stdout)

Optional
      -s, --seed=<int>          Starting point to the random generator (Default 0)
      -n, --nSymbols=<int>      Number of symbols generated (Default 100)
      -f, --frequency=<str>     The frequency of each base. It should be represented
                                in the following format: <fa,fc,fg,ft>.

Example: ./gto_genomic_gen_random_dna > output.seq

```

Output

The output of the `gto_genomic_gen_random_dna` program is a sequence group file which the synthetic DNA.

Using the input above with the seed value as 1 and the number of symbols as 400, an output example for this is the following:

```

TCTTTACTCGCGCGTTGGAGAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTTCATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACCTTCGGCGAGTCTGTGGAGTTTTTGTTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCCGAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTTCGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCCAAAGCCCACTGCGTCATGGAGCTCACGGTCTACATTTCATGACTGACTAACCGTAAACTGC

```

4.2 Program `gto_genomic_rand_seq_extra_chars`

The `gto_genomic_rand_seq_extra_chars` substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols. It works in sequence file formats.

For help type:

```
./gto_genomic_rand_seq_extra_chars -h
```

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

Input parameters

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

The attribution is given according to:

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

It substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols.
It works in sequence file formats

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

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

Example: ./gto_genomic_rand_seq_extra_chars < input.seq > output.seq
```

An example on such an input file is:

```
ANAAGACGNNNTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
NNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCCNNNNNGGAGAGGAAGCTCGGGAGNGTNNNGGCCAGGCGGCAGNNNNCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TANNNNCTACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGNNNAAGCAGCCTCCTGACTTTCTCGCTTGNNNNTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
NNATTACNNNCCTGNN
```

Output

The output of the `gto_genomic_rand_seq_extra_chars` program is a sequence file.

Using the input above, an output example for this is the following:

```
ATAAGACGGCTTCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
CTCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCCGACGGGAGAGGAAGCTCGGGAGTGTGTTGGCCAGGCGGCAGGAGACCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAATATCTCACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGCGGAAGCAGCCTCCTGACTTTCTCGCTTGGTTTTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
CGATTACGGCCCTGTC
```

4.3 Program `gto_genomic_dna_mutate`

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

For help type:

```
./gto_genomic_dna_mutate -h
```

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

Input parameters

The `gto_genomic_dna_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 sequence File.

The attribution is given according to:

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

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

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

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

Optional
      -s, --seed=<int>               Starting point to the random generator
      -m, --mutation-rate=<dbl>      Defines the mutation rate (default 0.0)
      -d, --deletion-rate=<dbl>      Defines the deletion rate (default 0.0)
      -i, --insertion-rate=<dbl>     Defines the insertion rate (default 0.0)
      -a, --ACGTN-alphabet           When active, the application uses the ACGTN alphabet

Example: ./gto_genomic_dna_mutate -s <seed> -m <mutation rate> -d <deletion rate> -i
<insertion rate> -a < input.seq > output.seq
```

An example on such an input file is:

```
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGC GGCTCTGTCTCCTTATGAAGTCAACAATTCGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGC GGCTCTGTCTCCTTATGAAGTCAACAATTCGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTCATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAGAACT
GTAATATATATAATGAACTTCGGCGAGTCTGTGGAGTTTTTGTGTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCCGAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCAAAAGCCCACTGCGTCATGGAGCTCACGGTCTACATTCATGACTGACTAACCGTAAACTGC
```

Output

The output of the `gto_genomic_dna_mutate` program is a sequence file with the synthetic mutation of input file.

Using the input above with the seed value as 1 and the mutation rate as 0.5, an output example for this is the following:

```
TCACGACTGTGCGGTTGGCACACCAGATAGGTGCTTCTACGTTTTGTATCTAATTTACAATTCTCGCTGGGAGTTCATTC
GCTATTGATGGGACTAGAAACCCATCCGTAGCTTGCCGCCGTTTAAAGAATAAACTCCACTTGCACCGAGACGTAGCGC
AACCAAGGCTATGTTCTTTGACCTTATGCGGTCCAACGCAGGAGTAGACCCCGTAGTTAGGTACTATCGCAGAATAGGC
TTAAGCAGCCGTGCTGAACGCTGGAGGGTCTGTTTAATTACTGAGTGAATGGAGAGCTAAGAGTTCGGAGCACCGCACGA
GGCTCAAGAGCGGAAGGGCGTCAGCCTGGCGACCACCTGCCTACCGCTCGAGTCTGTCTTCACTACAGTCCGTGGAGGAC
CCCCAACGACCTAGTATCCTACAAAGCCGCATACGACTTACAGAACAGGCTGTATCGTCAGGAGTGTGTACACGAAGAGT
A
```

4.4 Program gto_genomic_extract

The `gto_genomic_extract` extracts sequences from a sequence file, which the range is defined by the user in the parameters.

For help type:

```
./gto_genomic_extract -h
```

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

Input parameters

The `gto_genomic_extract` program needs two parameters, which defines the begin and the end of the extraction, and two streams for the computation, namely the input and output standard. The input stream is a sequence file.

The attribution is given according to:

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

It extracts sequences from a sequence file.

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

Basic options
-i, --init=<int>    The first position to start the extraction (default 0)
-e, --end=<int>     The last extract position (default 100)
< input.seq        Input sequence file (stdin)
> output.seq       Output sequence file (stdout)

Example: ./gto_genomic_extract -i <init> -e <end> < input.seq > output.seq
```

An example on such an input file is:

```
TCTTTACTCGCGCGTTGGAGAAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTCTATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACTTCGGCGAGTCTGTGGAGTTTTTGTGTCATTAGAGAGCCAAGAGGTGGGACGTCTCACGTA
```

```
GCCCCGAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTTCGACTCCTCCAGGAGC
TGAACAACCTTGTTCCTCCGGCAAAGCCCACTGCGTCATGGAGCTCACGGTCTACATTTCATGACTGACTAACCGTAAACTGC
```

Output

The output of the `gto_genomic_extract` program is a group sequence.

Using the input above with the value 0 as the extraction starting point and the 50 as the ending, an output example for this is the following:

```
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGC GGCTCTGTCTCCTTAT
```

4.5 Program `gto_genomic_period`

The `gto_genomic_period` calculates the best order depth of a sequence, using FCMs. It only works "ACGT", while the rest will be discarded.

This application has a dependency to represent the results. It requires the Gnuplot to show the execution result.

For help type:

```
./gto_genomic_period -h
```

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

Input parameters

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

The attribution is given according to:

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

It calculates the best order depth of a sequence, using FCMs. It only works "ACGT",
while the rest will be discarded.

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

Basic options
    < input.seq         Input sequence file format (stdin)
    > output             Output is given by log2(4)*K(x)/|x| (stdout)

Example: ./gto_genomic_period < input.seq > output
```

An example on such an input file is:


```
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGCGGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGCGGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTTCATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACTTCGGCGAGTCTGTGGAGTTTTTGTTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCCGAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCAAAGCCCCTGCGTCATGGAGCTCACGGTCTACATTGACTGACTAACCGTAAACTGC
```

Output

The output of the `gto_genomic_period` program is a execution report, followed by the plot with this information.

Using the input above, an report example for this is the following:

```
Running order: 1 ... Done!
Running order: 2 ... Done!
Running order: 3 ... Done!
Running order: 4 ... Done!
Running order: 5 ... Done!
Running order: 6 ... Done!
Running order: 7 ... Done!
Running order: 8 ... Done!
Running order: 9 ... Done!
Running order: 10 ... Done!
Running order: 11 ... Done!
Running order: 12 ... Done!
Running order: 13 ... Done!
Running order: 14 ... Done!
Running order: 15 ... Done!
Running order: 16 ... Done!
Running order: 17 ... Done!
Running order: 18 ... Done!
Running order: 19 ... Done!
Running order: 20 ... Done!
1 2.246
2 2.225
3 2.237
4 2.079
5 1.821
6 1.733
7 1.717
8 1.708
9 1.717
10 1.712
11 1.717
12 1.721
13 1.725
14 1.729
15 1.733
16 1.738
17 1.742
18 1.746
19 1.75
```

In the Figure 4.1 is represented the plot for the execution above.

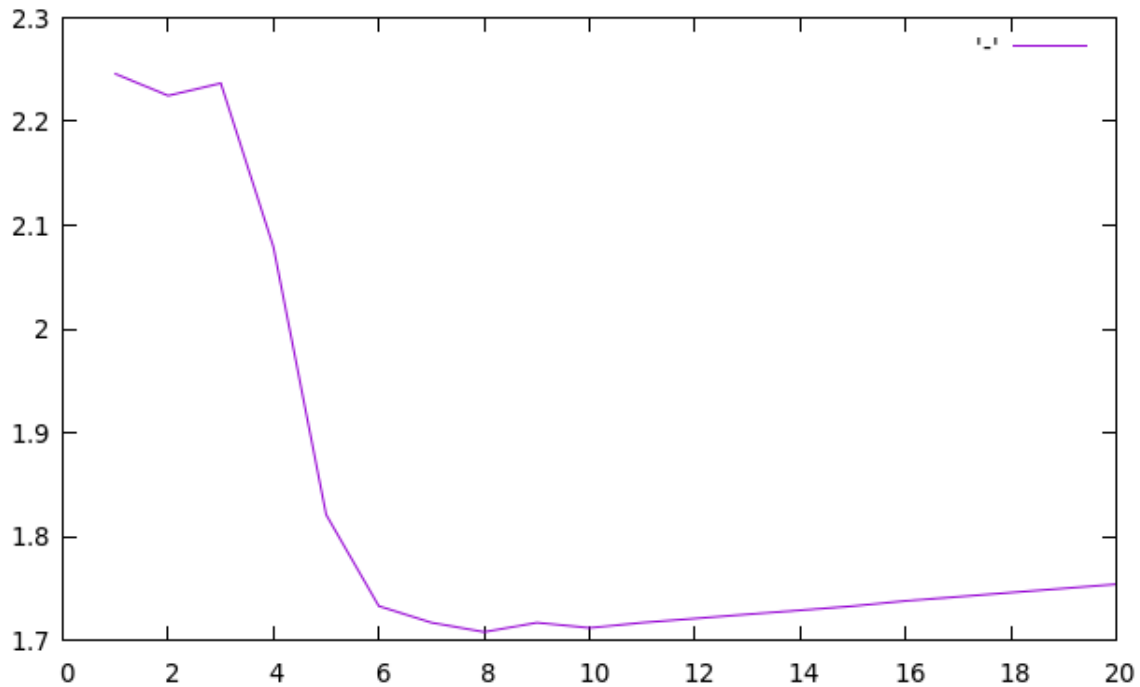


Figure 4.1: `gto_genomic_period` execution plot.

4.6 Program `gto_genomic_count_bases`

The `gto_genomic_count_bases` counts the number of bases in sequence, FASTA or FASTQ files. For help type:

```
./gto_genomic_count_bases -h
```

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

Input parameters

The `gto_genomic_count_bases` program needs two streams for the computation, namely the input and output standard. The input stream is a sequence, FASTA or FASTQ file.

The attribution is given according to:

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

```
It counts the number of bases in sequence, FASTA or FASTQ files.
```

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

```

Basic options
  < input      Input sequence, FASTA or FASTQ file format (stdin)
  > output     Output read information (stdout)

```

```

Example: ./gto_genomic_count_bases < input.seq > output

```

```

Output example :
File type       : value
Number of bases : value
Number of a/A   : value
Number of c/C   : value
Number of g/G   : value
Number of t/T   : value
Number of n/N   : value
Number of others : value

```

An example on such an input file is:

```

TCTTTACTCGCGCGTTGGAGAAATACAATAGTGCGGCTCTGTCTCCTTATGAAGTCAACAATTTTCGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGCGGCTCTGTCTCCTTATGAAGTCAACAATTTTCGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTTCATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACCTTCGGCGAGTCTGTGGAGTTTTTGTTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCGAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTTCGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCAAGCCCACTGCGTCATGGAGCTCACGGTCTACATTTCATGACTGACTAACCGTAAACTGC

```

Output

The output of the `gto_genomic_count_bases` program is report which describes the number of each base in the file, and the file type.

Using the input above, an output example for this is the following:

```

File type       : DNA
Number of bases : 480
Number of a/A   : 114
Number of c/C   : 116
Number of g/G   : 120
Number of t/T   : 130
Number of n/N   : 0
Number of others : 0

```

Chapter 5

Amino acid sequence tools

Current available amino acid sequence tools, for analysis and manipulation, are:

1. `gto_amino_acid_to_group`: it converts an amino acid sequence to a group sequence.
2. `gto_amino_acid_to_pseudo_dna`: it converts an amino acid (protein) sequence to a pseudo DNA sequence.

5.1 Program `gto_amino_acid_to_group`

The `gto_amino_acid_to_group` converts an amino acid sequence to a group sequence.

For help type:

```
./gto_amino_acid_to_group -h
```

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

Input parameters

The `gto_amino_acid_to_group` program needs two streams for the computation, namely the input and output standard. The input stream is an amino acid sequence. The attribution is given according to:

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

It converts a amino acid sequence to a group sequence.

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

Basic options
      < input.prot              Input amino acid sequence file (stdin)
      > output.group            Output group sequence file (stdout)

Example: ./gto_amino_acid_to_group < input.prot > output.group
Table:
Prot      Group
```

R	P	
H	P	Amino acids with electric charged side chains: POSITIVE
K	P	
-	-	
D	N	
E	N	Amino acids with electric charged side chains: NEGATIVE
-	-	
S	U	
T	U	
N	U	Amino acids with electric UNCHARGED side chains
Q	U	
-	-	
C	S	
U	S	
G	S	Special cases
P	S	
-	-	
A	H	
V	H	
I	H	
L	H	
M	H	Amino acids with hydrophobic side chains
F	H	
Y	H	
W	H	
-	-	
*	*	Others
X	X	Unknown

It can be used to group amino acids by properties, such as electric charge (positive and negative), uncharged side chains, hydrophobic side chains and special cases. An example on such an input file is:

```
IPFLKKQFALADKLVLKSLRQLLGGRIMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPNSIG
TLMPKAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

The output of the `gto_amino_acid_to_group` program is a group sequence.

Using the input above, an output example for this is the following:

```
HSHHHPPUHHHNNPHHHUPHPUHHSSPHPHSSSSHPHNSHHSHHHPHHSHUHPHSHSHUNUHHUHUSHPNHUHUSUUHS
UHHSPHNHPSNUUNHHHPSSHHPSPHPPSNNUHUUHUNNSHHPUSNHSNNHNNUSUHHHUNPHPNHHPUUUSPHHHSUH
HNUPHSPNPHHNUHHHHNNHPPHUUHHHSSSHNUHNNHHPUHUHPHPNPHNHHPUUNHHPHHN
```

5.2 Program `gto_amino_acid_to_pseudo_dna`

The `gto_amino_acid_to_pseudo_dna` converts an amino acid (protein) sequence to a pseudo DNA sequence.

For help type:

```
./gto_amino_acid_to_pseudo_dna -h
```

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

Input parameters

The `gto_amino_acid_to_pseudo_dna` program needs two streams for the computation, namely the input and output standard. The input stream is an amino acid sequence. The attribution is given according to:

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

It converts a protein sequence to a pseudo DNA sequence.

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

Basic options
    < input.prot        Input amino acid sequence file (stdin)
    > output.dna         Output DNA sequence file (stdout)

Example: ./gto_amino_acid_to_pseudo_dna < input.prot > output.dna
Table:
```

Prot	DNA
A	GCA
C	TGC
D	GAC
E	GAG
F	TTT
G	GGC
H	CAT
I	ATC
K	AAA
L	CTG
M	ATG
N	AAC
P	CCG
Q	CAG
R	CGT
S	TCT
T	ACG
V	GTA
W	TGG
Y	TAC
*	TAG
X	GGG

It can be used to generate pseudo-DNA with characteristics passed by amino acid (protein) sequences. An example on such an input file is:

```
IPFLLKKQFALADKLVL SKLRQLLGGR IKMMP CGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPN SIG
TLMPKAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

The output of the `gto_amino_acid_to_pseudo_dna` program is a DNA sequence.

Using the input above, an output example for this is the following:

```
ATCCCGTTTCTGCTGAAAAACAGTTTGC ACTGGCAGACAACTGGTACTGTCTAAACTGCGTCAGCTGCTGGGCGGCCG
TATCAAAATGATGCCGTGCGGCGGCGCAAAACTGGAGCCGGCAATCGGCCTGTTTTTTCATGCAATCGGCATCAACATCA
AACTGGGCTACGGCATGACGGAGACGACGGCAACGGTATCTTGCTGGCATGACTTTCAGTTTAACCCGAAC TCTATCGGC
ACGCTGATGCCGAAAGCAGAGGTAAAAATCGGCGAGACAACGAGATCCTGGTACGTGGCGGCATGGTAATGAAAGGCTA
CTACAAAAAACCGGAGGAGACGGCACAGGCATTTACGGAGGACGGCTTTCTGAAAACGGGCGACGCAGGCGAGTTTGACG
AGCAGGGCAACCTGTTTATCACGGACCGTATCAAAAGAGCTGATGAAAACGTCTAACGGCAAATACATCGCACCGCAGTAC
ATCGAGTCTAAAATCGGCAAAGACAAATTTATCGAGCAGATCGCAATCATCGCAGACGCAAAAAATACGTATCTGCACT
GATCGTACCGTGCTTTGACTCTCTGGAGGAGTACGCAAAACAGCTGAACATCAAATACCATGACCGTCTGGAGCTGCTGA
AAAACTCTGACATCCTGAAAAATGTTTGAG
```

Chapter 6

General purpose tools

1. `gto_char_to_line`: it splits a sequence into lines, creating an output sequence which has a char for each line.
2. `gto_reverse`: it reverses the order of a sequence.
3. `gto_new_line_on_new_x`: it splits different rows with a new empty row.
4. `gto_upper_bound`: it sets an upper bound in a file with a value per line.
5. `gto_lower_bound`: it sets an lower bound in a file with a value per line.
6. `gto_brute_force_string`: it generates all combinations, line by line, for an inputted alphabet and specific size.
7. `gto_real_to_binary_with_threshold`: it converts a sequence of real numbers into a binary sequence, given a threshold.
8. `gto_sum`: it adds decimal values in file, line by line, splitted by spaces or tabs.
9. `gto_filter`: it filters numerical sequences.
10. `gto_word_search`: it search for a word in a file.
11. `gto_permute_by_blocks`: it permutates by block sequence, FASTA and Multi-FASTA files.
12. `gto_info`: it gives the basic properties of the file, namely size, cardinality, distribution percentage of the symbols, among others.
13. `gto_segment`: it segments a filtered sequence.
14. `gto_comparative_map`: it creates a visualization for comparative maps.
15. `gto_xs`: it ...
16. `gto_geco`: it ...
17. `gto_gede`: it ...

6.1 Program gto_char_to_line

The `gto_char_to_line` splits a sequence into lines, creating an output sequence which has a char for each line.

For help type:

```
./gto_char_to_line -h
```

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

Input parameters

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

The attribution is given according to:

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

It splits a sequence into lines, creating an output sequence which has a char for each line.

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

Basic options

```
< input.seq        Input sequence file (stdin)
> output.seq       Output sequence file (stdout)
```

```
Example: ./gto_char_to_line < input.seq > output.seq
```

An example on such an input file is:

```
ACAAGACGGGCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

Output

The output of the `gto_char_to_line` program is a group sequence splited by `\n` foreach character. Using the input above, an output example for this is the following:

```
A
C
A
A
G
A
C
G
G
C
C
T
C
C
T
G
C
T
G
C
T
...
```

6.2 Program `gto_reverse`

The `gto_reverse` reverses the order of a sequence file.

For help type:

```
./gto_reverse -h
```

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

Input parameters

The `gto_reverse` program needs two streams for the computation, namely the input and output standard.

The input stream is a sequence file.

The attribution is given according to:

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

It reverses the order of a sequence file.

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

Basic options

```
< input.seq        Input sequence file (stdin)
> output.seq       Output sequence file (stdout)
```

```
Example: ./gto_reverse < input.seq > output.seq
```

An example on such an input file is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

Output

The output of the `gto_reverse` program is a group sequence.

Using the input above, an output example for this is the following:

```
AAGTCCAGACATTAATTTGAACGCACTCGTAAGTACCCACTCCAAAATAAACGTCTCCTCTTCCAGAAGGTCTTCTTCA
AGGACGTCCCGTAAGACAGGGCCGCGGCCTAACGACCCCCCACGCGGAAGGACGGCGGACCGGTGGAGGGCTCGAAGG
AGAGGATACTCCCCGGGCGGTGACCGGACCTCCAGGTGAGTTTGGTGGTTTCGCTCCTTTCAGTCCTCCGACGAAAAGGA
ATAAGGACGGCGGAAGGACGTATACGAGCGACAGAGCCGGCCACCCCGGTGGGAGGTCCCCGTCCCGTCGCCACCGGCACC
GGGGCCTCTCGTCGTCGTCCTCCTCCGGCCCCCTGTTACCGTAGAACAAAGTCCAGACATTAATTTGAACGCACTCGTAA
GTACCCACTCCAAAATCGACCCCCCACCTCTTCCAGAAGGTCTTCTTCAAGGACGTCCCGAAACGTCTCTAAGACAGG
GCCGCGCGCCTAAGCGCGGTGACCGGACGAAGGACGGCGGACCGGTGGAGGGCTCGAAGGAGAGGATACTCCCCGGGCCT
CCAGGTGAGTTTGGTGAAGGACGGCGGAAGGACGTATACGAGCGACAGAGCCGGGTTCGCTCCTTTCAGTCCTCCGACGAA
AAGGAATCCACCCCGGCCCTGTTACCGTCGTCCTCGCCACCTGGGAGGTCCCGGCACCGGGGCCTCTCGTCGTCGTC
GTCCTCCGGCAGAAACA
```

6.3 Program `gto_new_line_on_new_x`

The `gto_new_line_on_new_x` splits different rows with a new empty row.

For help type:

```
./gto_new_line_on_new_x -h
```

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

Input parameters

The `gto_new_line_on_new_x` program needs two streams for the computation, namely the input and output standard. The input stream is a matrix file format with 3 columns.

The attribution is given according to:

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

It splits different rows with a new empty row.

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

Basic options

```
< input          Input file with 3 column matrix format (stdin)
> output          Output file with 3 column matrix format (stdout)
```

Example: `./gto_new_line_on_new_x < input > output`

An example on such an input file is:

```
1  2  2
1  2  2
4  4  1
10 12  2
15 15  1
45 47  3
45 47  3
45 47  3
45 47  3
55 55  1
```

Output

The output of the `gto_new_line_on_new_x` program is a 3 column matrix, with an empty line between different rows.

Using the input above, an output example for this is the following:

```
1.000000    2.000000    2.000000
1.000000    2.000000    2.000000

4.000000    4.000000    1.000000

10.000000   12.000000    2.000000

15.000000   15.000000    1.000000

45.000000   47.000000    3.000000
45.000000   47.000000    3.000000
45.000000   47.000000    3.000000
45.000000   47.000000    3.000000

55.000000   55.000000    1.000000
```

6.4 Program gto_upper_bound

The `gto_upper_bound` sets an upper bound in a file with a value per line.

For help type:

```
./gto_upper_bound -h
```

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

Input parameters

The `gto_upper_bound` program needs two streams for the computation, namely the input and output standard. The input stream is a numeric file.

The attribution is given according to:

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

It sets an upper bound in a file with a value per line.

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

Basic options
    -u, --upperbound=<int>    The upper bound value
    < input                    Input numeric file (stdin)
    > output                    Output numeric file (stdout)

Example: ./gto_upper_bound -u <upperbound> < input > output
```

An example on such an input file is:

```
0.123
3.432
2.341
1.323
7.538
4.122
0.242
0.654
5.633
```

Output

The output of the `gto_upper_bound` program is a set of numbers truncated at the a defined upper bound. Using the input above, an output example for this is the following:

```
Using upper bound: 4
0.123000
3.432000
2.341000
```

```
1.323000
4.000000
4.000000
0.242000
0.654000
4.000000
```

6.5 Program `gto_lower_bound`

The `gto_lower_bound` sets an lower bound in a file with a value per line.

For help type:

```
./gto_lower_bound -h
```

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

Input parameters

The `gto_lower_bound` program needs two streams for the computation, namely the input and output standard. The input stream is a numeric file.

The attribution is given according to:

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

It sets an lower bound in a file with a value per line.

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

Basic options
    -u, --lowerbound=<int>    The lower bound value
    < input                    Input numeric file (stdin)
    > output                    Output numeric file (stdout)

Example: ./gto_lower_bound -u <lowerbound> < input > output
```

An example on such an input file is:

```
0.123
3.432
2.341
1.323
7.538
4.122
0.242
0.654
5.633
```

Output

The output of the `gto_lower_bound` program is a set of numbers truncated at the a defined lower bound. Using the input above, an output example for this is the following:

```
Using lower bound: 2
2.000000
3.432000
2.341000
2.000000
7.538000
4.122000
2.000000
2.000000
5.633000
```

6.6 Program `gto_brute_force_string`

The `gto_brute_force_string` generates all combinations, line by line, for an inputted alphabet and specific size.

For help type:

```
./gto_brute_force_string -h
```

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

Input parameters

The `gto_brute_force_string` program needs some paramenters for the computation, namely the alphabet and the key size.

The attribution is given according to:

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

It generates all combinations, line by line, for an inputted alphabet and specific size.

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

Basic options
-a, --alphabet=<str> The input alphabet
-s, --size=<int>     The combinations size
> output            Output all the combinations (stdout)

Example: ./gto_brute_force_string -a <alphabet> -s <size> > output
```

Output

The output of the `gto_brute_force_string` program is a set of all possible word combinations with a defined size, using the input alphabet.

Using the input above with the alphabet "abAB" with the word size of 3, an output example for this is the following:

```
aaa
aab
aaA
aaB
aba
...
BBb
BBA
BBB
```

6.7 Program `gto_real_to_binary_with_threshold`

The `gto_real_to_binary_with_threshold` converts a sequence of real numbers into a binary sequence, given a threshold. The numbers below to the threshold will be 0.

For help type:

```
./gto_real_to_binary_with_threshold -h
```

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

Input parameters

The `gto_real_to_binary_with_threshold` program needs two streams for the computation, namely the real sequence as input. These numbers should be splitted by lines.

The attribution is given according to:

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

It converts a sequence of real numbers into a binary sequence given a threshold.

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

Basic options

```
-t, --threshold=<dbl>    The threshold in real format
< input.num              Input numeric file (stdin)
> output.bin              Output binary file (stdout)
```

```
Example: ./gto_real_to_binary_with_threshold -t <threshold> < input.num > output.bin
```

An example on such an input file is:


```
12.25
1.2
5.44
5.51
7.97
2.34
8.123
```

Output

The output of the `gto_real_to_binary_with_threshold` program is a binary sequence.

Using the input above with the threshold of 5.5, an output example for this is the following:

```
1
0
0
1
1
0
1
```

6.8 Program `gto_sum`

The `gto_sum` adds decimal values in file, line by line, splitted by spaces or tabs.

For help type:

```
./gto_sum -h
```

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

Input parameters

The `gto_sum` program needs program needs two streams for the computation, namely the input, which is a decimal file.

The attribution is given according to:

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

It adds decimal values in file, line by line, splitted by spaces or tabs.

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

Basic options

```
< input.num        Input numeric file (stdin)
> output.num       Output numeric file (stdout)
```

Optional

```
-r, --sumrows      When active, the application adds all the values line by line
-a, --sumall       When active, the application adds all values
```

```
Example: ./gto_sum -a < input.num > output.num
```

An example on such an input file is:

```
0.123   5   5
3.432
2.341   3   2
1.323
7.538   5
4.122
0.242
0.654
5.633   10
```

Output

The output of the `gto_sum` program is a sum of the elements in the input file.

Executing the application with the provided input and with the flag to add only the elements in each row, the output of this execution is:

```
10.123000
3.432000
7.341000
1.323000
12.538000
4.122000
0.242000
0.654000
15.633000
```

6.9 Program `gto_filter`

The `gto_filter` filters numerical sequences using a low-pass filter.

For help type:

```
./gto_filter -h
```

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

Input parameters

The `gto_filter` program needs two streams for the computation, namely the input and output standard.

The input stream is a numeric file.

The attribution is given according to:

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

It filters numerical sequences using a low-pass filter.

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

Basic options

```
< input.num              Input numeric file (stdin)
> output.num             Output numeric file (stdout)
```

Optional

```
-w, --windowsize=<int>   Window size (default 0)
-d, --drop=<int>         Discard elements (default 0.0)
-t, --windowtype=<int>   Window type (0=Hamm, 1=Hann, 2=Black, 3=rec) (default 0 (Hamm))
-c, --onecolumn          Read from one column
-p, --printone           Print one column
-r, --reverse            Reverse mode
```

```
Example: ./gto_filter -w <windowsize> -d <drop> -t <windowtype> -c -p -r < input.num > output.num
```

An example on such an input file is:

```
1    1.77
5    2.18
10   2.32
15   3.15
20   2.52
25   4.43
30   1.23
```

Output

The output of the `gto_filter` program is a numeric file, identical of the input.

Using the input above with the window size of 3, an output example for this is the following:

```
Got 7 entries from file
1    2.085
5    2.256
10   2.507
15   2.757
20   2.905
25   2.860
30   2.674
```

6.10 Program `gto_word_search`

The `gto_word_search` search for a word in a file. It is case sensitive.

For help type:

```
./gto_word_search -h
```

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

Input parameters

The `gto_word_search` program needs two streams for the computation, namely the input and output standard. The input stream is a text file.

The attribution is given according to:

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

Searching for a word in a text file. It is case sensitive.

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

Basic options
      -w, --word=<str>    Word to search in the file
      < input.txt         Input text file (stdin)
      > output.txt        Output text file (stdout)

Example: ./gto_word_search -w <word> < input.txt > output.txt
```

An example on such an input file is:

```
No guts, no story. Chris Brady
My life is my message. Mahatma Gandhi
Screw it, let's do it. Richard Branson
Boldness be my friend. William Shakespeare
Keep going. Be all in. Bryan Hutchinson
My life is my argument. Albert Schweitzer
Fight till the last gasp. William Shakespeare
Leave no stone unturned. Euripides
```

Output

The output of the `gto_word_search` program is a text file with the matching paragraphs and the location of the word found.

Using the input above with the word "Shakespeare", an output example for this is the following:

```
Found match in range [ 1536 : 2048 ]
Boldness be my friend. William Shakespeare

Found match in range [ 3072 : 3584 ]
Fight till the last gasp. William Shakespeare
```

6.11 Program gto_permute_by_blocks

The `gto_permute_by_blocks` permutes by block sequence, FASTA and Multi-FASTA files. For help type:

```
./gto_ -h
```

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

Input parameters

The `gto_permute_by_blocks` program needs two streams for the computation, namely the input and output standard. The input stream is a sequence, FASTA or Multi-FASTA file.

The attribution is given according to:

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

It permutes by block sequence, FASTA and Multi-FASTA files.

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

Basic options
-b, --numbases=<int> The number of bases in each block
-s, --seed=<int>     Starting point to the random generator
< input              Input sequence, FASTA or Multi-FASTA file format (stdin)
> output             Output sequence, FASTA or Multi-FASTA file format (stdout)

Example: ./gto_permute_by_blocks -b <numbases> -s <seed> < input.fasta > output.fasta
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCGCAACACGCAAGTTTAATTCGCAAGTTAGACCTGAACGGGAGGTGGCCACGCAAGTT
```

Output

The output of the `gto_permute_by_blocks` program is a sequence, FASTA or Multi-FASTA file permuted following some parameters.

Using the input above with the base number as 80, an output example for this is the following:

```
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
```

```
TAAACCTCACCCATGAATGCTCGCAACACGCAAGTTTAATTGCGCAAGTTAGACCTGAACGGGAGGTGGCCACGCAAGTT
```

6.12 Program gto_info

The `gto_info` gives the basic properties of the file, namely size, cardinality, distribution percentage of the symbols, among others.

For help type:

```
./gto_info -h
```

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

Input parameters

The `gto_info` program needs two streams for the computation, namely the input and output standard. The input stream is a file without any specific format.

The attribution is given according to:

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

It gives the basic properties of the file, namely size, cardinality, distribution
percentage of the symbols, among others.

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

Basic options
    < input          Input file (stdin)
    > output          Output read information (stdout)

Optional
    -a, --ascii      When active, the application shows the ASCII codes

Example: ./gto_info < input > output

Output example :
Number of symbols : value
Alphabet size     : value
Alphabet         : value
Symbol distribution:
<Symbol/Code ASCII> <Symbol count> <Distribution percentage>
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
```

```
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAAC TTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCGCAACACGCAAGTTTAATTGCAAGTTAGACCTGAACGGGAGGTGGCCACGCAAGTT
```

Output

The output of the `gto_info` program is a set of information related to the file read.

Using the input above, an output example for this is the following:

```
Number of symbols : 453
Alphabet size      : 28
Alphabet           : |srponmiedcaTRNHGCBA>=6420 \n
Symbol distribution:
| : 2      0.4415011
s : 3      0.66225166
r : 1      0.22075055
p : 1      0.22075055
o : 2      0.4415011
n : 1      0.22075055
m : 2      0.4415011
i : 1      0.22075055
e : 2      0.4415011
d : 1      0.22075055
c : 3      0.66225166
a : 2      0.4415011
T : 66     14.569536
R : 1      0.22075055
N : 1      0.22075055
H : 1      0.22075055
G : 117    25.827815
C : 131    28.918322
B : 2      0.4415011
A : 89     19.646799
> : 1      0.22075055
= : 2      0.4415011
6 : 2      0.4415011
4 : 2      0.4415011
2 : 2      0.4415011
0 : 6      1.3245033
  : 4      0.88300221
\n : 5     1.1037528
```

6.13 Program `gto_segment`

The `gto_segment` segments a filtered sequence.

For help type:

```
./gto_segment -h
```

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

Input parameters

The `gto_segment` program needs two streams for the computation, namely the input and output standard.

The input stream is a numeric file.

The attribution is given according to:

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

It segments a filtered sequence.

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

Basic options
      -t, --threshold=<dbl>    The segment threshold
      < input.num              Input numeric file (stdin)
      > output                  Output the segment file (stdout)

Example: ./gto_segment -t <threshold> < input.num > output
```

An example on such an input file is:

```
1    1.77
5    2.18
10   2.32
15   3.15
20   2.52
25   4.43
30   1.23
```

Output

The output of the `gto_segment` program is the interval of values “below the threshold.

Using the input above with a threshold of 3, an output example for this is the following:

```
0:10
```

6.14 Program `gto_comparative_map`

The `gto_comparative_map` creates a visualization for comparative maps.

For help type:

```
./gto_comparative_map -h
```

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

Input parameters

The `gto_comparative_map` program needs an input file with the plot positions, respecting a defined structure.

The attribution is given according to:

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

It creates a visualization for comparative maps.

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

Basic options
<FILE>              Contigs filename with positions (.pos),

Optional

-h                  Give this help,
-V                  Display version number,
-v                  Verbose mode (more information),
-l <link>           Link type between maps [0;4],
-w <width>          Chromosome width,
-s <space>          Space between chromosomes,
-m <mult>           Color id multiplication factor,
-b <begin>          Color id beggining,
-c <minimum>        Minimum block size to consider,
-i                  Do NOT show inversion maps,
-r                  Do NOT show regular maps,
-o <FILE>           Output image filename with map,

Example: ./gto_comparative_map -o map.svg map.config
```

The input file needs to have the following structure:

```
todo
```

An example on such an input file is:

```
#SCF      5000000 5000000
aaa       1      1000000 1      1000000 bbbb      3000000 4000000 3000000 4000000
bbb       1500000 2000000 1500000 2000000 cccc      1500000 2000000 1500000 2000000
aaa       2000000 3000000 2000000 3000000 bbbb      3000000 2000000 3000000 2000000
```

Output

The output of the `gto_comparative_map` program is a executing report, and a svg plot with the maps. Using the input above, an output example for this is the following:

```

==[ PROCESSING ]=====
Printing plot ...
Found 2 regular regions.
Found 1 inverted regions.
Done!

==[ STATISTICS ]=====
Total cpu time: 0 second(s).

```

In the Figure 6.1 is represented the plot for the execution above.

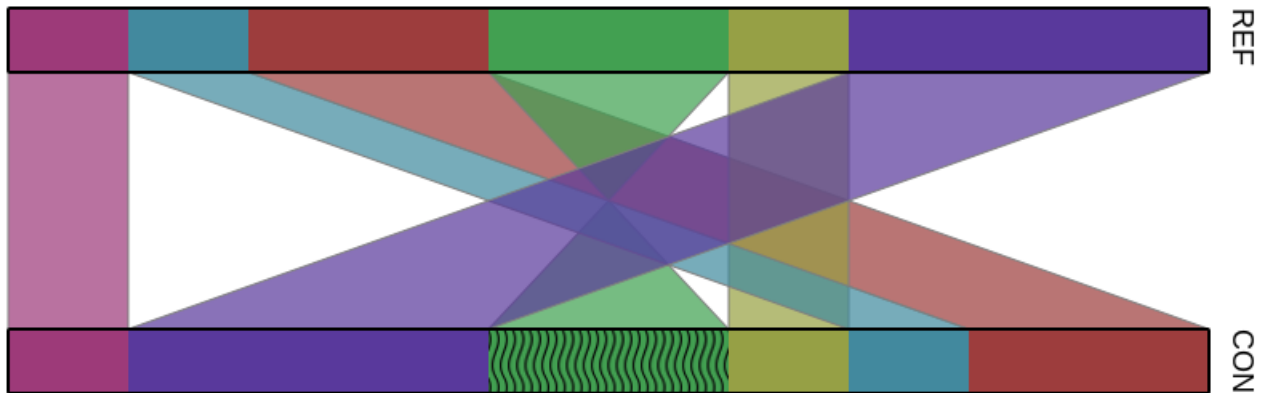


Figure 6.1: `gto_comparative_map` execution plot.

6.15 Program `gto_xs`

The `gto_xs` .

For help type:

```
./gto_xs -h
```

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

Input parameters

The `gto_xs` program needs program needs two streams for the computation, namely the input and output standard. ...

The attribution is given according to:

```
TODO
```

An example on such an input file is:

```
TO DO
```

Output

The output of the `gto_xs` program is ...

Using the input above, an output example for this is the following:

```
TO DO
```

6.16 Program `gto_geco`

The `gto_geco` .

For help type:

```
./gto_geco -h
```

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

Input parameters

The `gto_geco` program needs program needs two streams for the computation, namely the input and output standard. ...

The attribution is given according to:

```
TODO
```

An example on such an input file is:

```
TO DO
```

Output

The output of the `gto_geco` program is ...

Using the input above, an output example for this is the following:

```
TO DO
```

6.17 Program `gto_gede`

The `gto_gede` .

For help type:

```
./gto_gede -h
```

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

Input parameters

The `gto_gede` program needs two streams for the computation, namely the input and output standard. ...

The attribution is given according to:

```
TODO
```

An example on such an input file is:

```
TO DO
```

Output

The output of the `gto_gede` program is ...

Using the input above, an output example for this is the following:

```
TO DO
```