



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	3
1.1	Installation	3
1.2	License	4
2	Amino acid sequence tools	5
2.1	Program gto_amino_acid_to_group	5
2.2	Program gto_amino_acid_to_pseudo_dna	6
3	FASTQ tools	9
3.1	Program gto_fastq_to_fasta	9
3.2	Program gto_fastq_to_mfasta	10
3.3	Program gto_fastq_exclude_n	12
3.4	Program gto_fastq_extract_quality_scores	13
3.5	Program gto_fastq_info	14
3.6	Program gto_fastq_maximum_read_size	15
3.7	Program gto_fastq_minimum_quality_score	16
3.8	Program gto_fastq_minimum_read_size	17
3.9	Program gto_fastq_rand_extra_chars	19
3.10	Program gto_fastq_from_seq	20
3.11	Program gto_fastq_mutate	22
4	FASTA tools	24
4.1	Program gto_fasta_to_seq	24
4.2	Program gto_fasta_from_seq	26
4.3	Program gto_fasta_extract	27
4.4	Program gto_fasta_extract_by_read	28
4.5	Program gto_fasta_info	29
4.6	Program gto_fasta_mutate	30
4.7	Program gto_fasta_rand_extra_chars	32
4.8	Program gto_fasta_extract_read_by_pattern	33
4.9	Program gto_fasta_find_n_pos	34

4.10	Program gto_fasta_split_reads	35
4.11	Program gto_fasta_rename_human_headers	37
5	Genomic sequence tools	39
5.1	Program gto_genomic_gen_random_dna	39
5.2	Program gto_genomic_rand_seq_extra_chars	40
6	General purpose tools	42
6.1	Program gto_char_to_line	42
6.2	Program gto_reverse	43
6.3	Program gto_new_line_on_new_x	45
	Bibliography	46

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 [1]. The storage of genomic data is being addressed using specific file formats, such as FASTQ and FASTA. Therefore, its analysis and manipulation is crucial [2]. Many frameworks for analysis and manipulation emerged, namely GALAXY [3], GATK [4], HTSeq [5], MEGA [6], 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:

```
https://pratas.github.io/GTO
```

1.1 Installation

For **GTO** installation, run:

```
git clone https://github.com/pratas/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

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.

2.1 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
TLMPPAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

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

An example, for the input, is:

```
HSHHHPPUHHHHNPHHHUPHPUHHSSPHPHSSSSHPHNSHHSHHHPHSHUHPHSHSHUNUUHUUHUSHPNHUHUSUUHS
UHHSPHNHPHSNUUNHHHPSSHHHPSSHPPSNNUHUHHUNNSHHPUSNHSNHNNUUUHHUNPHPNHHPUUUSPHHHSUH
HNUPHSPNPHNUHHHHHNPPHHUHHHHSSHNUHNNHHPUHUHPHPNPHNHHPUUNHHHPHN
```

2.2 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 SKLRQLLGGRIMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPNSIG
TLMPKAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

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

An example, for the input, is:

```
ATCCCGTTTCTGCTGAAAAACAGTTTGCACCTGGCAGACAAACTGGTACTGTCTAAACTGCGTCAGCTGCTGGGCGGCCG
TATCAAAATGATGCCGTGCGGCGGCGCAAACTGGAGCCGGCAATCGGCCTGTTTTTCATGCAATCGGCATCAACATCA
AACTGGGCTACGGCATGACGGAGACGACGGCAACGGTATCTTGCTGGCATGACTTTCAGTTTAACCCGAACCTCTATCGGC
ACGCTGATGCCGAAAGCAGAGGTAAAAATCGGCGAGAACACGAGATCCTGGTACGTGGCGGCATGGTAATGAAAGGCTA
CTACAAAAAACCGGAGGAGACGGCACAGGCATTTACGGAGGACGGCTTTCTGAAAACGGGCGACGCAGGCGAGTTTGACG
AGCAGGGCAACCTGTTTATCACGGACCGTATCAAAGAGCTGATGAAAACGTCTAACGGCAAATACATCGCACCGCAGTAC
ATCGAGTCTAAAATCGGCAAAGACAAATTTATCGAGCAGATCGCAATCATCGCAGACGCAAAAAATACGTATCTGCACT
GATCGTACCGTGCTTTGACTCTCTGGAGGAGTACGCAAAACAGCTGAACATCAAATACCATGACCGTCTGGAGCTGCTGA
AAAACTCTGACATCCTGAAAAATGTTTGAG
```

Chapter 3

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 informations 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.

3.1 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 parameters.

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
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+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 is a FASTA file.

An example, for the input, is:

```
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
```

3.2 Program `gto_fastq_to_mfasta`

The `gto_fastq_to_mfasta` converts 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 parameters.

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
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
G TTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+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 is a Multi-FASTA file.

An example, for the input, is:

```
>SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
>SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
G TTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
```

3.3 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 paramters.

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              Output read information (stdout)
```

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

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
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTTAAGGGTTNTCAAATAGA
+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
NTTCAGGGATACGACGNTTGTATTTTAAAGATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-1)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.

Using the max value as 5, an example for this input, is:

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

3.4 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 paramters.

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            Output read information (stdout)

Example: ./gto_fastq_extract_quality_scores < input.fastq > output

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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGCTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT

[illegible]

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.

An example, for the input, is:

```

IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
Total reads      : 2
Total Quality-Scores : 144

```

3.5 Program gto_fastq_info

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

For help type:

```
./gto_fastq_info -h
```

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

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 informations 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
```

```

It analyses the basic informations 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

```

```
-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
```

```
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
```

```
< 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
```

```
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

```

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

```

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

Output

The output of the `gto_fastq_info` program is a set of informations related with the file readed.

An example, for the input, is:

```
Total reads      : 2
Max read length  : 72
Min read length  : 72
Min QS value     : 41
Max QS value     : 73
QS range         : 33
```

3.6 Program `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 paramters.

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              Output read information (stdout)
```

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

Output example :

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

[illegible]

The output of the `gto_fastq_maximum_read_size` program is a set of all the filtered FASTQ reads, followed by the execution report.

Using the size value as 60, an example for this input, is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
Total reads      : 2
Filtered reads   : 1
```

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
```

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                  Output read information (stdout)

Example: ./gto_fastq_minimum_quality_score < input.fastq > output

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
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
GTTTCAGGGATACGACGCTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
54599<>77977==6=?I6IBI::33344235521677999>>><<@@A@BB CDGGBFFH>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 minimum average value as 30, an example for this input, is:

```

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

```

3.8 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                   Output read information (stdout)

Example: ./gto_fastq_minimum_read_size < input.fastq > output

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_minimum_read_size` program is a set of all the filtered FASTQ reads, followed by the execution report.

Using the size value as 65, an example for this input, is:

Output

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

An example, for the input, is:

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

3.10 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:

```

ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTACCCATGAATGCTCAGCAAGTT
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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq2
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq3
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq4
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq5
TAAAACCTACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq6
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACGGCCGAGACAGCGAGCATATGCA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq7
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq8
GGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SeqToFastq9
AGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTACCCATGAATGCTCAGCAAGTT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

```

```
@SeqToFastq10
TAATTACAGACCTGAA
+
FFFFFFFFFFFFFFFFFF
```

3.11 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
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_mutate` program is a FASTQ file with the synthetic mutation of input file. Using the seed value as 1 and the mutation rate as 0.5, an example for this input, is:

```

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

```


Chapter 4

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.

4.1 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGCGCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

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

An example, for the input, is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGCGCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
```

```
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA
```

4.2 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:

```
ACAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCAGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAAGTGGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA
```

Output

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

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

```
>SeqToFasta
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCCTCCTGCTG
CTGCTGCTCTCCGGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA
```

4.3 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAAGTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

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

An example, using the value 0 as extraction starting point and the 50 as the end, for the provided input, is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGG
```

4.4 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 parameters.

Input parameters

The `gto_fasta_extract_by_read` 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 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
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

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

An example, using the value 0 as extraction starting point and the 50 as the end, for the provided input, is:

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

4.5 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA

```

Output

The output of the `gto_fasta_info` program is a set of informations related with the file readed.

An example, for the input, is:

```

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

```

4.6 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAAGTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGCGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAAGTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```


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 seed value as 1 and the edition rate as 0.5, an example for this input, is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACGCAACGNATTCTGCTGATCATANTGTNCCGCNCCCCNCGCAGGGGNCCTCNCNNGCACACATNGTACCATTGTCCAC
NCTTNCANGTNANCGCTAGCAGGCTACNGTTTNTCCTCNCCTANNCCAANCNGGCGTNNNTACTGTCAGTGCAGGCA
TNGGTCCGCGNCGNCCCTCCGNAACGGCACCGGAGACGAAGCTCGGNGGNTATACAGGTGTCANGAAACATCCCCGCGNC
GNGTGNCCNNGAANCCANAGAGTATCTCACTCACAAACCTGCGTGCACNTCTAGAGNANGACCTTACNCACNTCCCNNT
NNGTACCACACCAATGAACGCTGCAGAAAGTCTGTTTNNAGGNGNGCA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ATTTGAAGGCAANCGGNCCAGNAATNCGGNGGGTGCNGCTCNTGTNGGCTACGGNCATCGCGGCCCTGCTNTANTAAGCN
TGAACCACCGNTCGNNGCACTTAGCAATNGCGNAANCCGTCGGCACGGCGGAGACNAANCCGCTANTNNTTTCCCGCTNA
ATGGNTGTACAAGACCNACTANACCANCCCTCCGTCAACCACTGGAGCGCANGATGGNNGCTGNCTAGNAGNCNNTGAG
GCGCTCCNTCCTANAAANCCGTGGNCGAGCNCCCTATGGNAGNGTGGGGGTTTTACCGGAAGACCNTCGNGCCCTATGGG
AGCAATCANAANCTAGAAAGCTTACNGATGGTGANGAANTAGACTANG
```

4.7 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
ANAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGAGGGTCCNCCGCTGCCCTGCTGCCATTGNCNCC
NGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGGCAGGAA
GNGGTTTGAGTGGACCTCCNGGCCCTCATAGGAGAGGAAGCNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNC
GCGAATCCGNGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCENN
TAAANNNTACCCATGAATGCTCACGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGNGCGCGGGACAGAATCTCCTTCTCCACCCCCCENNNTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACC
NGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGG
ANAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGCNCAGGGTCCNCCGCTGCCCTGCTGCCATTGN
GAGGAAGCNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNCNGGTTTGAGTGGACCTCCNGGCCCTCATAGGA
TCACGCAANTTTAATTACAGACCTGAATAAANNNTACCCATGAATGC
```

Output

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

An example, for the input, is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ATAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCCCCGCTGCCCTGCTGCCATTGTCCCC
TGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGGCAGGAA
GAGGTTTGAGTGGACCTCCCGGCCCTCATAGGAGAGGAAGCCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGTC
GCGAATCCGGGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCTTG
TAAAAGATCACCCATGAATGCTCACGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGTGCGCGGGACAGAATCTCCTTCTCCACCCCCCATCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACC
GGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGG
AGAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGTCCCTGGCTCCAGGGTCTCCGCTGCCCTGCTGCCATTGC
GAGGAAGCGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCGCGGTTTGAGTGGACCTCCTGGCCCTCATAGGA
TCACGCAACTTTAATTACAGACCTGAATAAATGTCACCCATGAATGC
```

4.8 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

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

An example, using the pattern "264", for the provided input, is:

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

4.9 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
NCNNNACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGTTCCACCGCTGCCCTGCTGCCATTGTCCCC
GNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTNGTTTGGAGTGACCTCCGGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
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.

An example, for the input, is:

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

4.10 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

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.

An example, for the input, is:

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

4.11 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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAAGTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
> AB000263 | acc = AB000263 | descr = Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGACCTCCCAGGCCAGTGCCGGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAAGTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

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

An example, for the input, is:

```
>chr1
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGTAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA

>chr2
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGTAGTGGACCTCCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Chapter 5

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.

5.1 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)
-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 with the synthetic DNA.

Using the seed value as 1 and the number of symbols as 400, an example of an execution, is:

```
TCTTTACTCGCGCGTTGGAGAAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTCGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTCGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTCTCATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACTTCGGCGAGTCTGTGGAGTTTTTGTGTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCCAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTTCGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCAAAGCCCCTGCGTCATGGAGCTCACGGTCTACATTTCATGACTGACTAACCGTAAACTGC
```

5.2 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
NNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCNNNNGGAGAGGAAGCTCGGGAGNGTNNNGGCCAGGCGGCAGNNNNCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TANNNNCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGNNNAAGCAGCCTCCTGACTTTCCTCGCTTGNNNNTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
NNATTACNNNCCTGNN
```

Output

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

An example, for the input, is:

```
ATAAGACGGCTTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
CTCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCGACCGGGAGAGGAAGCTCGGGAGTGTGTTGGCCAGGCGGCAGGAGACCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAATATCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGCGGAAGCAGCCTCCTGACTTTCCTCGCTTGTTTTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
CGATTACGGCCCTGTC
```

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.

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:

ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGAACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA

Output

The output of the `gto_char_to_line` program is a group sequence splited by `\n` foreach character.

An example, for the input, is:

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:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGCGGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

Output

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

An example, for the input, is:

```
AAGTCCAGACATTAATTTGAACGCACTCGTAAGTACCCACTCCAAAATAAACGTCCTCCTCTTCCAGAAGGTCTTCTTCA
AGGACGTCCCGTAAGACAGGGCCGCGGCCTAACGACCCCCCACGCGGAAGGACGGCGGACCGGTGGAGGGCTCGAAGG
AGAGGATACTCCCCGGGCCGTGACCGGACCCCTCCAGGTGAGTTTGGTGGTTCGCTCCTTTCAGTCCTCCGACGAAAAGGA
ATAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGCCACCCGGTGGGAGGTCCCGTCCCGTCGCCACCGGCACC
GGGGCCTCTCGTCGTCGTCCTCCTCGGCCCTTGTTACCGTAGAACAAAGTCCAGACATTAATTTGAACGCACTCGTAA
GTACCCACTCCAAAATCGACCCCCCACCTCTTCCAGAAGGTCTTCTTCAAGGACGTCCCGAAACGTCTCTAAGACAGG
GCCGCGCGCCTAAGCGCGGTGACCGGACGAAGGACGGCGGACCGGTGGAGGGCTCGAAGGAGAGGATACTCCCCGGGCCT
CCAGGTGAGTTTGGTGAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGTTCGCTCCTTTCAGTCCTCCGACGAA
AAGGAATCCACCCGGGCCCTGTTACCGTCGTCGCGCACCTGGGAGGTCCCGGCACCGGGGCCTCTCGTCGTCGTC
```

```
GTCTCCGGCAGAACA
```

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.

An example, for the input, is:

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

Bibliography

- [1] E. R. Mardis, “DNA sequencing technologies: 2006–2016,” *Nature protocols*, vol. 12, no. 2, p. 213, 2017.
- [2] H. Buermans and J. Den Dunnen, “Next generation sequencing technology: advances and applications,” *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, vol. 1842, no. 10, pp. 1932–1941, 2014.
- [3] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor *et al.*, “Galaxy: a platform for interactive large-scale genome analysis,” *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.
- [4] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. Del Angel, M. A. Rivas, M. Hanna *et al.*, “A framework for variation discovery and genotyping using next-generation dna sequencing data,” *Nature genetics*, vol. 43, no. 5, pp. 491–498, 2011.
- [5] S. Anders, P. T. Pyl, and W. Huber, “Htseq—a python framework to work with high-throughput sequencing data,” *Bioinformatics*, p. btu638, 2014.
- [6] S. Kumar, G. Stecher, and K. Tamura, “Mega7: Molecular evolutionary genetics analysis version 7.0 for bigger datasets,” *Molecular Biology and Evolution*, p. msw054, 2016.