



## A toolkit for DNA sequence analysis and manipulation

D. Pratas (pratas@ua.pt)

J. R. Almeida (joao.rafael.almeida@ua.pt)

A. J. Pinho (ap@ua.pt)

IEETA/DETI, University of Aveiro, Portugal

Version 1.7.17

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Installation . . . . .	2
1.2	License . . . . .	2
<b>2</b>	<b>FASTQ tools</b>	<b>4</b>
2.1	Program goose-fastq2fasta . . . . .	4
2.2	Program goose-fastq2mfasta . . . . .	5
2.3	Program goose-FastqExcludeN . . . . .	7
2.4	Program goose-FastqExtractQualityScores . . . . .	8
2.5	Program goose-FastqInfo . . . . .	9
2.6	Program goose-FastqMaximumReadSize . . . . .	10
2.7	Program goose-FastqMinimumQualityScore . . . . .	11
2.8	Program goose-FastqMinimumReadSize . . . . .	12
2.9	Program goose-mutatefastq . . . . .	14
2.10	Program goose-randfastqextrachars . . . . .	15
2.11	Program goose-seq2fastq . . . . .	16
<b>3</b>	<b>FASTA tools</b>	<b>19</b>
3.1	Program goose-fasta2seq . . . . .	19
3.2	Program goose-fastaextract . . . . .	21
3.3	Program goose-fastaextractbyread . . . . .	22
3.4	Program goose-fastainfo . . . . .	23
3.5	Program goose-mutatefasta . . . . .	24
3.6	Program goose-randfastaextrachars . . . . .	26
3.7	Program goose-extractreadbypattern . . . . .	27
3.8	Program goose-findnpos . . . . .	28
3.9	Program goose-seq2fasta . . . . .	29
3.10	Program goose-splitreads . . . . .	31
<b>4</b>	<b>Genomic sequence tools</b>	<b>33</b>
4.1	Program goose-genrandommdna . . . . .	33

4.2	Program goose-randseqextrachars . . . . .	34
<b>5</b>	<b>Amino acid sequence tools</b>	<b>36</b>
5.1	Program goose-AminoAcidToGroup . . . . .	36
5.2	Program goose-ProteinToPseudoDNA . . . . .	37
<b>6</b>	<b>General purpose tools</b>	<b>40</b>
6.1	Program goose-reverse . . . . .	40
6.2	Program goose-newlineonnewx . . . . .	41
	<b>Bibliography</b>	<b>42</b>

# 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. Its storage have been made using specific file formats, such as FASTQ and FASTA. Therefore, its analysis and manipulation is crucial [?]. Several frameworks for analysis and manipulation emerged, namely **GALAXY** [?], **GATK** [?], **HTSeq** [?], **MEGA** [?], among others. In the majority, these frameworks require licenses and do not provide a low level access to the information, since they are commonly approached by scripting or interfaces.

We describe **GOOSE**, a (free) novel toolkit for analyzing and manipulating FASTA-FASTQ formats and sequences (DNA, amino acids, text), with many complementary tools. The toolkit is for Linux-based systems, built for fast processing. **GOOSE** supports pipes for easy integration. It includes tools for information display, randomizing, edition, conversion, extraction, searching, calculation and visualization. **GOOSE** is prepared to deal with very large datasets, typically in the scale Gigabytes or Terabytes.

The toolkit is a command line version, using the prefix “goose-” followed by the suffix with the respective name of the program. **GOOSE** is implemented in C language and it is available, under GPLv3, at:

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

### 1.1 Installation

For **GOOSE** installation, run:

```
git clone https://github.com/pratas/goose.git
cd goose/src/
make
```

### 1.2 License

The license is **GPLv3**. In resume, everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. For details on the license, consult: <http://www.gnu.org/>

[licenses/gpl-3.0.html](#).

## Chapter 2

# FASTQ tools

Current available tools for FASTQ format analysis and manipulation include:

1. `goose-fastq2fasta`: it converts a FASTQ file format to a pseudo FASTA file.
2. `goose-fastq2mfasta`: it converts a FASTQ file format to a pseudo Multi-FASTA file.
3. `goose-FastqExcludeN`: it discards the FASTQ reads with the minimum number of "N" symbols.
4. `goose-FastqExtractQualityScores`: it extracts all the quality-scores from FASTQ reads.
5. `goose-FastqInfo`: it analyses the basic informations of FASTQ file format.
6. `goose-FastqMaximumReadSize`: it filters the FASTQ reads with the length higher than the value defined.
7. `goose-FastqMinimumQualityScore`: it discards reads with average quality-score below of the defined.
8. `goose-FastqMinimumReadSize`: it filters the FASTQ reads with the length smaller than the value defined.
9. `goose-randfastqextrachars`: it substitutes in the FASTQ files, the DNA sequence the outside ACGT chars by random ACGT symbols.
10. `goose-seq2fastq`: it converts a genomic sequence to pseudo FASTQ file format.
11. `goose-mutatefastq`: it creates a synthetic mutation of a FASTQ file given specific rates of mutations, deletions and additions.

### 2.1 Program `goose-fastq2fasta`

The `goose-fastq2fasta` 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:

```
./goose-fastq2fasta -h
```

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

## Input parameters

The `goose-fastq2fasta` 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: ./goose-fastq2fasta [options] [--] args]
       or: ./goose-fastq2fasta [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: ./goose-fastq2fasta < 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
GTTTCAGGGATACGACGCTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

## Output

The output of the `goose-fastq2fasta` program is a FASTA file.

An example, for the input, is:

```
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
GTTTCAGGGATACGACGCTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
```

## 2.2 Program `goose-fastq2mfasta`

The `goose-fastq2mfasta` 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:

```
./goose-fastq2mfasta -h
```

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

## Input parameters

The `goose-fastq2mfasta` 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: ./goose-fastq2mfasta [options] [--] args]
       or: ./goose-fastq2mfasta [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: ./goose-fastq2mfasta < 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
GTTTCAGGGATACGACGTTTGTATTTTAAAGTCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I
```

## Output

The output of the `goose-fastq2mfasta` 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
GTTTCAGGGATACGACGTTTGTATTTTAAAGTCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
```



## 2.3 Program `goose-FastqExcludeN`

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

For help type:

```
./goose-FastqExcludeN -h
```

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

### Input parameters

The `goose-FastqExcludeN` 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: ./goose-FastqExcludeN [options] [--] args]
or: ./goose-FastqExcludeN [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: ./goose-FastqExcludeN < 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
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACCTAAGGGTTNTCAAATAGA
+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
NTTCAGGGATACGACGNTTGTATTTTAAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-1)8I
```

### Output

The output of the `goose-FastqExcludeN` 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:

[illegible]

## 2.4 Program `goose-FastqExtractQualityScores`

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

For help type:

```
./goose -FastqExtractQualityScores -h
```

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

## Input parameters

The `goose-FastqExtractQualityScores` 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: ./goose-FastqExtractQualityScores [options] [--] args]
or: ./goose-FastqExtractQualityScores [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: ./goose-FastqExtractQualityScores < 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  
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA  
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72  
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/  
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72  
GTTCAGGGATACGACGTTTTGTATTTTAAGAATCTGAAGCAGAAAGTCGATGATAATACGCGTCGTTTTATCAT
```

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

## Output

The output of the `goose-FastqExtractQualityScores` program is a set of all the quality scores from the FASTQ reads, followed by the execution report.

An example, for the input, is:

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

## 2.5 Program `goose-FastqInfo`

The `goose-FastqInfo` analyses the basic informations of FASTQ file format.

For help type:

```
./goose-FastqInfo -h
```

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

### Input parameters

The `goose-FastqInfo` 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: ./goose-FastqInfo [options] [--] args]
or: ./goose-FastqInfo [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: ./goose-FastqInfo < 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

## Output

The output of the `goose-FastqInfo` 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
```

## 2.6 Program `goose-FastqMaximumReadSize`

The `goose-FastqMaximumReadSize` filters the FASTQ reads with the length higher than the value defined.

For help type:

```
./goose-FastqMaximumReadSize -h
```

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

### Input parameters

The `goose-FastqMaximumReadSize` 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: ./goose-FastqMaximumReadSize [options] [--] args]
or: ./goose-FastqMaximumReadSize [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: `./goose-FastqMaximumReadSize < 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 `goose-FastqMaximumReadSize` 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
GGGTGATGGCCGCTGCCGATGGCGTCAAAATCCCACCAAGTTACCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
Total reads      : 2
Filtered reads   : 1
```

## 2.7 Program `goose-FastqMinimumQualityScore`

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

```
./goose-FastqMinimumQualityScore -h
```

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

### Input parameters

The `goose-FastqMinimumQualityScore` 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: ./goose-FastqMinimumQualityScore [options] [--] args]
       or: ./goose-FastqMinimumQualityScore [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: ./goose-FastqMinimumQualityScore < 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
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
GTTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+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 `goose-FastqMinimumQualityScore` 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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCCTTAACAACCTTAAGGGTTTTCAAATAGA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
Total reads      : 2
Filtered reads   : 1

```

## 2.8 Program `goose-FastqMinimumReadSize`

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

```
./goose-FastqMinimumReadSize -h
```

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

## Input parameters

The `goose-FastqMinimumReadSize` 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: ./goose-FastqMinimumReadSize [options] [--] args]
      or: ./goose-FastqMinimumReadSize [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: ./goose-FastqMinimumReadSize < 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 `goose-FastqMinimumReadSize` 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:

```
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72  
GTTCAGGGATACGACGTTTTGTATTTAAGAATCTGAAGCAGAAAGTCGATGATAATACGCCTCGTTTTATCAT  
+  
IIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I  
Total reads      : 2  
Filtered reads   : 1
```

## 2.9 Program goose-mutatefastq

The `goose-mutatefastq` 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:

```
./goose-mutatefastq -h
```

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

## Input parameters

The `goose-mutatefastq` 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: ./goose-mutatefastq [options] [--] args]
    or: ./goose-mutatefastq [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: ./goose-mutatefastq -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
GGGTGATGGCGCTGCCGATGGCGTCAAAATCCCACCAAGTTACCCTTAACAACCTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

## Output

The output of the `goose-mutatefastq` 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
GGACTTTGAGGTGTGGCGATAGACTGAAAACACTTCAGGGTAAAATCACTCGCAAAAGTGCTATGGTTATGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>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
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII>IIIII-I)8I

```

## 2.10 Program `goose-randfastqextrachars`

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

For help type:

```
./goose-randfastqextrachars -h
```

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

### Input parameters

The `goose-randfastqextrachars` 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: ./goose-randfastqextrachars [options] [--] args]
or: ./goose-randfastqextrachars [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: ./goose-randfastqextrachars < 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 `goose-randfastqextrachars` 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
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
```

## 2.11 Program `goose-seq2fastq`

The `goose-seq2fastq` converts a genomic sequence to pseudo FASTQ file format.

For help type:

```
./goose-seq2fastq -h
```

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

### Input parameters

The `goose-seq2fastq` 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: ./goose-seq2fastq [options] [--] args]
or: ./goose-seq2fastq [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: ./goose-seq2fastq -l <lineSize> -n <name> < input.seq > output.fastq

```

An example on such an input file is:

```

ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA

```

## Output

The output of the `goose-seq2fastq` program is a pseudo FASTQ file.

An example, using the size line as 80 and the read's header as "Seq2Fastq", for the input, is:

```

@Seq2Fastq1
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq2
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq3
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq4
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq5
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq6
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA

```

```

+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq7
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq8
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq9
AGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@Seq2Fastq10
TAATTACAGACCTGAA
+
FFFFFFFFFFFFFFFFFFFF

```

## Chapter 3

# FASTA tools

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

1. `goose-fasta2seq`: it converts a FASTA or Multi-FASTA file format to a seq.
2. `goose-fastaextract`: it extracts sequences from a FASTA file, which the range is defined by the user in the parameters.
3. `goose-fastaextractbyread`: 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.
4. `goose-fastainfo`: it shows the readed information of a FASTA or Multi-FASTA file format.
5. `goose-mutatefasta`: it reates a synthetic mutation of a fasta file given specific rates of editions, deletions and additions.
6. `goose-randfastaextrachars`: it substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols.
7. `goose-extractreadbypattern`: it extracts reads from a Multi-FASTA file format given a pattern in the header.
8. `goose-findnpos`: it reports the "N" regions in a sequence or FASTA (seq) file.
9. `goose-seq2fasta`: it converts a genomic sequence to pseudo FASTA file format.
10. `goose-splitreads`: it splits a Multi-FASTA file to multiple FASTA files.

### 3.1 Program `goose-fasta2seq`

The `goose-fasta2seq` converts a FASTA or Multi-FASTA file format to a sequence.

For help type:

```
./goose-fasta2seq -h
```

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

## Input parameters

The `goose-fasta2seq` 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: ./goose-fasta2seq [options] [--] args]
      or: ./goose-fasta2seq [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: ./goose-fasta2seq < input.fasta > output.seq
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

## Output

The output of the `goose-fasta2seq` program is a group sequence.

An example, for the input, is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGGACCTCCAGGCCAGTGCCG
```

```

GCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCTCTGCAATAAAACCTCACCCATGAATGCTCAGCAAGTT
TAATTACAGACCTGAA

```

## 3.2 Program goose-fastextract

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

For help type:

```
./goose-fastextract -h
```

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

### Input parameters

The `goose-fastextract` 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: ./goose-fastextract [options] [--] args]
       or: ./goose-fastextract [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: ./goose-fastextract -i <init> -e <end> < input.fasta > output.seq

```

An example on such an input file is:

```

>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGTCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAA

```

### Output

The output of the `goose-fastextract` 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
```

### 3.3 Program goose-fastaextractbyread

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

For help type:

```
./goose-fastaextractbyread -h
```

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

#### Input parameters

The `goose-fastaextractbyread` 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: ./goose-fastaextractbyread [options] [--] args
or: ./goose-fastaextractbyread [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: ./goose-fastaextractbyread -i <init> -e <end> < input.fasta > output.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
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCCTGGAGGGT
GGCCCCACCGCGCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCTGACTTTCTCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAA
```



```
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

## Output

The output of the `goose-fastextractbyread` 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
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCC
```

## 3.4 Program `goose-fastainfo`

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

```
./goose-fastainfo -h
```

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

### Input parameters

The `goose-fastainfo` 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: ./goose-fastainfo [options] [--] args]
or: ./goose-fastainfo [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: ./goose-fastainfo < 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
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGCGCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

## Output

The output of the `goose-fastainfo` 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
```

## 3.5 Program `goose-mutatefasta`

The `goose-mutatefasta` 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:

```
./goose-mutatefasta -h
```

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

### Input parameters

The `goose-mutatefasta` 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: ./goose-mutatefasta [options] [--] args]
      or: ./goose-mutatefasta [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: ./goose-mutatefasta -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
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGTTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAACCTCACCCATGAATGCTCAGCAAGTTTAATTACAGACCTGAA

```

## Output

The output of the `goose-mutatefasta` 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
NCTTNCANGTNANCGCTAGCAGGCTACNGTTTNTCCTCNCCATANNCCAANCNGGCGTNNNTACACTGGCACGTGCAGGCA
TNGGTGGCNGGNNCCTCCGGNAACGGCACCGGAGACGAAGCTCGGNGGNTATACAGGTGTCANGAAACATCCCCGCGNC
GNGTGNCNNNGAANCCANAGAGTATCTCACTCACAACCTGCGGTGCACNTCTAGAGNANGACCTTACNCACNTCCCNNT
NNGTACCACACCAATGAACGCTGCAGAAAGTCTGTTTTNAGGNGNGCA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ATTTGAAGGCAANCNGNCCAGNAATNCGNGGGTGCGCTCNTGTNGGCTACGGNCATCGGGCCCTGCTNTANTAAGCN

```

```
TGAACCACCGNTCGNNGCACTTAGCAATNGCGNAANCCGTCGGCACGGCGGAGACNAANCCGCTANTNNTTCCCGCTNA
ATGGNTGTACAAGACCNACTANACCANCCTCCGTCACCACACTGGAGCGCANGATGGNNCGCTGNCTAGNAGNCNNTGAG
GCGCTCCNTCCTANAAANCCGTGGNCGAGCNCCCTATGGNAGNGTGGGGGTTTTACCGGAAGACCNTCGNGCCCTATGGG
AGCAATCANAANCTAGAAAGCTTACNGATGGTGANGAANTAGACTANG
```

### 3.6 Program goose-randfastaextrachars

The `goose-randfastaextrachars` 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:

```
./goose-randfastaextrachars -h
```

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

#### Input parameters

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

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ANAAGACGGCCTCCTGCTGCTGCTCCTCGGGGCCACGNCCCTGGAGGGTCCNCCGCTGCCCTGCTGCCATTGNCNCC
NGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGGCAGGAA
GNGGTTTGAAGTGGACCTCCNGGCCCTCATAGGAGAGGAAGCNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNC
GCGAATCCGNGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCN
TAAANNNTACCCATGAATGCTCAGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGNGCGCGGGACAGAATCTCCTTCTCCACCCCCCNNTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACC
NGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGG
ANAAGACGGCCTCCTGCTGCTGCTCCTCGGGGCCACGNCCCTGGCNCAGGGTCCNCCGCTGCCCTGCTGCCATTGN
```

```
GAGGAAGC NNGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNCGNGGTTTGAGTGACCTCCNGGCCCCCTCATAGGA
TCACGCAANTTTAATTACAGACCTGAATAAANNNTCACCCATGAATGC
```

## Output

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

An example, for the input, is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ATAAGACGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCCCCGCTGCCCTGCTGCCATTGTCCCC
TGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGGCAGGAA
GAGGTTTGAGTGACCTCCCGGCCCTCATAGGAGAGGAAGCCGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGTC
GCGAATCCGGGCGCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCTTG
TAAAAGATCACCCATGAATGCTCACGCAAAATTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGTGCGCCGGGACAGAATCTCTTCTCCACCCCCCATCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACC
GGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGG
AGAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGTCCCTGGCTCCAGGGTCTCCGCTGCCCTGCTGCCATTGC
GAGGAAGCGGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCGCGGTTTGAGTGACCTCCTGGCCCCCTCATAGGA
TCACGCAACTTTAATTACAGACCTGAATAAAATGTCACCCATGAATGC
```

## 3.7 Program `goose-extractreadbypattern`

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

For help type:

```
./goose-extractreadbypattern -h
```

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

### Input parameters

The `goose-extractreadbypattern` 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: ./goose-extractreadbypattern [options] [--] args]
or: ./goose-extractreadbypattern [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: ./goose-extractreadbypattern -p <pattern> < input.fasta > output.fasta
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCCTCGCTTG
GTGGTTTGAGTGGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

## Output

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

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

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

## 3.8 Program goose-findnpos

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

For help type:

```
./goose-findnpos -h
```

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

### Input parameters

The `goose-findnpos` 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: ./goose-findnpos [options] [--] args]
or: ./goose-findnpos [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: ./goose-findnpos < 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
NCNNNACGGCCTCCTGCTGCTGCTGCTCCTCGGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTNGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACNTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAN

```

## Output

The output of the **goose-findnpos** 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

```

## 3.9 Program goose-seq2fasta

The **goose-seq2fasta** converts a genomic sequence to pseudo FASTA file format.

For help type:

```
./goose-seq2fasta -h
```

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

### Input parameters

The **goose-seq2fasta** 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: ./goose-seq2fasta [options] [--] args]
or: ./goose-seq2fasta [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: `./goose-seq2fasta -l <lineSize> -n <name> < input.seq > output.fasta`

An example on such an input file is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCCTGCCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

## Output

The output of the `goose-seq2fasta` program is a pseudo FASTA file.

An example, using the size line as 80 and the read's header as "Seq2Fasta", for the input, is:

```
>Seq2Fasta
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCCTGCCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTTGAGTGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```



## 3.10 Program goose-splitreads

The **goose-splitreads** splits a Multi-FASTA file to multiple FASTA files.

For help type:

```
./goose-splitreads -h
```

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

### Input parameters

The **goose-splitreads** 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: ./goose-splitreads [options] [--] args]
      or: ./goose-splitreads [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: ./goose-splitreads < input.fasta
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCTGCTGCTGCTGCTCCTCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCGGGACAGAATCTCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCGGCCTCTGCTGCTGCTGCTCCTCGGGGCCACGGCCACCGCTGCCCTGCCCTGGAGGGT
GGCCCCACCGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCTGACTTTCTCGCTTG
GTGGTTTGAGTGACCTCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCAGCAATCCGCGCGCGGGACAGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCTCTCTGCAAA
TAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

### Output

The output of the **goose-splitreads** 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
```

## Chapter 4

# Genomic sequence tools

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

1. `goose-genrandomdna`: it generates a synthetic DNA.
2. `goose-randseqextrachars`: it substitutes in the DNA sequence the outside ACGT chars by random ACGT symbols.

### 4.1 Program `goose-genrandomdna`

The `goose-genrandomdna` generates a synthetic DNA.

For help type:

```
./goose-genrandomdna -h
```

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

#### Input parameters

The `goose-genrandomdna` program needs one stream for the computation, namely the output standard.

The attribution is given according to:

```
Usage: ./goose-genrandomdna [options] [--] args]
or: ./goose-genrandomdna [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: ./goose-genrandomdna > output.seq
```

## Output

The output of the `goose-genrandomdna` 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:

```
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
TCTTTACTCGCGCGTTGGAGAAATACAATAGTGGCGCTCTGTCTCCTTATGAAGTCAACAATTTGCTGGGACTTGCGGC
GACTTCATCGTGGTCTCTGTCTATTATGCGCTCCAACGCATAACTTTGCGCCAGAAGATAGATAGAATGGTGTAAAGAACT
GTAATATATATAATGAACCTTCGGCGAGTCTGTGGAGTTTTTGTGTCATTAGAGAGCCAAGAGGTCGGACGTCCTCACGTA
GCCCAGACGGGCAGGGCGATGGCGACTGAACGGGCTCCATATCACTTTGAGCTTTTATGCTTTTCGACTCCTCCAGGAGC
TGAACAACCTTGTTCCCGGCAAAGCCCACTGCGTCATGGAGCTCACGGTCTACATTCATGACTGACTAACCGTAAACTGC
```

## 4.2 Program goose-randseqextrachars

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

For help type:

```
./goose-randseqextrachars -h
```

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

### Input parameters

The `goose-randseqextrachars` 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: ./goose-randseqextrachars [options] [--] args]
or: ./goose-randseqextrachars [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: ./goose-randseqextrachars < input.seq > output.seq
```

An example on such an input file is:

```
ANAAGACGNNNTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
NNCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCNNGGAGAGGAAGCTCGGGAGNGTNNNGGCCAGGCGGCAGNNNNCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TANNNNCTACCCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGNNNAAGCAGCCTCCTGACTTTCCTCGCTTGNNNNTTGTAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
NNATTACNNNCCTGNN
```

## Output

The output of the `goose-randseqextrachars` program is a sequence file.

An example, for the input, is:

```
ATAAGACGGCTTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
CTCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGACCTCCGGGGCCCGACCGGGAGAGGAAGCTCGGGAGTGTGTTGGCCAGGCGGCAGGAGACCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCACCCCCCAGC
TAATATCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGCGGAAGCAGCCTCCTGACTTTCCTCGCTTGGTTTTTTGTAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
CGATTACGGCCCTGTC
```

## Chapter 5

# Amino acid sequence tools

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

1. `goose-AminoAcidToGroup`: it converts an amino acid sequence to a group sequence.
2. `goose-ProteinToPseudoDNA`: it converts an amino acid (protein) sequence to a pseudo DNA sequence.

### 5.1 Program `goose-AminoAcidToGroup`

The `goose-AminoAcidToGroup` converts an amino acid sequence to a group sequence.

For help type:

```
./goose-AminoAcidToGroup -h
```

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

#### Input parameters

The `goose-AminoAcidToGroup` 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: ./goose-AminoAcidToGroup [options] [--] args]
or: ./goose-AminoAcidToGroup [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: ./goose-AminoAcidToGroup < 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:

```

IPFLKKQFALADKLVL SKLRQLLGGR IKMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPN SIG
TLMPKAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE

```

## Output

The output of the `goose-AminoAcidToGroup` program is a group sequence.

An example, for the input, is:

```

HSHHHPPUHHHHNPHHHUPHPUHHSSPHPHSSSSHPHNSHHSHHHPHSHUHPHSHSHUNUHUHUSHPNHUHUSUHS
UHHSPhNHPhSNUUNHHHPSSHHHP SHHPPSNNUHUHHUNNSHHPPUSNHSNNHNNUSUHHHUNPHPNHHPUUUSPHHHSUH
HNUPHSPNPHHN UHHHHHNNPHPHUHHHHSSHNUNHNNHPPUHUHPHPNPHNHHPUUNHHPHHN

```

## 5.2 Program `goose-ProteinToPseudoDNA`

The `goose-ProteinToPseudoDNA` converts an amino acid (protein) sequence to a pseudo DNA sequence.

For help type:

```
./goose-ProteinToPseudoDNA -h
```

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

## Input parameters

The `goose-ProteinToPseudoDNA` 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: ./goose-ProteinToPseudoDNA [options] [--] args]
       or: ./goose-ProteinToPseudoDNA [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: ./goose-ProteinToPseudoDNA < 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:

```
IPFLLKKQFALADKLVLSKLRQLLGGRICKMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPNSIG
TLMPKAEVKIGENNEILVRGGMVMKGYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```



## Output

The output of the `goose-ProteinToPseudoDNA` program is a DNA sequence.

An example, for the input, is:

```
ATCCCGTTTCTGCTGAAAAACAGTTTGC ACTGGCAGACAAACTGGTACTGTCTAAACTGCGTCAGCTGCTGGGCGGCCG
TATCAAAATGATGCCGTGCGGCGGCGCAAAACTGGAGCCGGCAATCGGCCTGTTTTTTCATGCAATCGGCATCAACATCA
AACTGGGCTACGGCATGACGGAGACGACGGCAACGGTATCTTGCTGGCATGACTTTCAGTTTAACCCGAACCTCTATCGGC
ACGCTGATGCCGAAAGCAGAGGTAAAAATCGGCGAGAACACGAGATCCTGGTACGTGGCGGCATGGTAATGAAAGGCTA
CTACAAAAAACCGAGGAGACGGCACAGGCATTTACGGAGGACGGCTTTCTGAAACGGGCGACGCAGGCGAGTTTGACG
AGCAGGGCAACCTGTTTATCACGGACCGTATCAAAGAGCTGATGAAAACGTCTAACGGCAAATACATCGCACCGCAGTAC
ATCGAGTCTAAATCGGCAAAGACAAATTTATCGAGCAGATCGCAATCATCGCAGACGCAAAAAATACGTATCTGCACT
GATCGTACCGTGCTTTGACTCTCTGGAGGAGTACGCAAAACAGCTGAACATCAAATACCATGACCGTCTGGAGCTGCTGA
AAAACTCTGACATCCTGAAAAATGTTTGAG
```

# Chapter 6

## General purpose tools

1. `goose-char2line`: it splits a sequence into lines, creating an output sequence which has a char for each line.
2. `goose-reverse`: it reverses the order of a sequence.
3. `goose-newlineonnewx`: it splits different rows with a new empty row.

### 6.1 Program `goose-char2line`

The `goose-char2line` splits a sequence into lines, creating an output sequence which has a char for each line.

For help type:

```
./goose-char2line -h
```

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

#### Input parameters

The `goose-char2line` 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: ./goose-char2line [options] [--] args]
or: ./goose-char2line [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: ./goose-char2line < input.seq > output.seq
```

An example on such an input file is:

ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC  
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA  
GTGGTTTGAGTGACCTCCGGGGCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC  
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCAGC  
TAAACCTCACCCATGAATGCTCAGCAAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCTCCTGCTG  
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCCTGGAGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA  
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG  
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC  
AGAATGCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAAATAAACCTCACCCATGAATGCTCAGCAAGTT  
TAATTACAGACCTGAA

## Output

The output of the `goose-char2line` program is a group sequence splited by `\n` foreach character.

An example, for the input, is:

## 6.2 Program goose-reverse

The **goose-reverse** reverses the order of a sequence file.

For help type:

```
./goose-reverse -h
```

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

## Input parameters

The **goose-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: ./goose-reverse [options] [--] args]
       or: ./goose-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: ./goose-reverse < input.seq > output.seq
```

An example on such an input file is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCACCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAAATTACAGACCTGAAACAAGATGCCATTGTCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCGCCTGGAGGGTGGCCCCACCGCGGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCTCGCTTGGTGGTTGAGTGGACCTCCAGGCCAGTGCCG
GGCCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

## Output

The output of the **goose-reverse** program is a group sequence.

An example, for the input, is:

```
AAGTCCAGACATTAATTTGAACGCACTCGTAAGTACCCACTCCAAAATAAACGTCTCCTCTTCCAGAAGGTCTTCTTCA
AGGACGTCCCGTAAGACAGGGCCGCGCGCCTAACGACCCCCCACGCGGAAGGACGGCGGACCGGTGGAGGGCTCGAAGG
AGAGGATACTCCCCGGGCGGTGACCGGACCTCCAGGTGAGTTTGGTGGTTTCGCTCCTTTTCAGTCCTCCGACGAAAAGGA
ATAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGCCACCCGGTGGGAGGTCCCGTCCCGTCGCCACCGGCACC
GGGGCCTCTCGTCGTCGTCCTCCTCCGGCCCCCTGTTACCGTAGAACAAAGTCCAGACATTAATTTGAACGCACTCGTAA
GTACCCACTCCAAAATCGACCCCCCACCTCTTCCAGAAGGTCTTCTTCAAGGACGTCCCGAAACGTCTCTAAGACAGG
GCCGCGCGCCTAAGCGCGGTGACCGGACGAAGGACGGCGGACCGGTGGAGGGCTCGAAGGAGAGGATACTCCCCGGGCT
CCAGGTGAGTTTGGTGAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGTTTCGCTCCTTTTCAGTCCTCCGACGAA
AAGGAATCCACCCGGCCCCCTGTTACCGTCGTCCTCGCCACCTGGGAGGTCCCGGCACCGGGGCCTCTCGTCGTCGTC
```

```
GTCTCCGGCAGAACA
```

## 6.3 Program goose-newlineonnewx

The `goose-newlineonnewx` splits different rows with a new empty row.

For help type:

```
./goose-newlineonnewx -h
```

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

### Input parameters

The `goose-newlineonnewx` 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: ./goose-newlineonnewx [options] [--] args]
      or: ./goose-newlineonnewx [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: ./goose-newlineonnewx < 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 `goose-newlineonnewx` 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