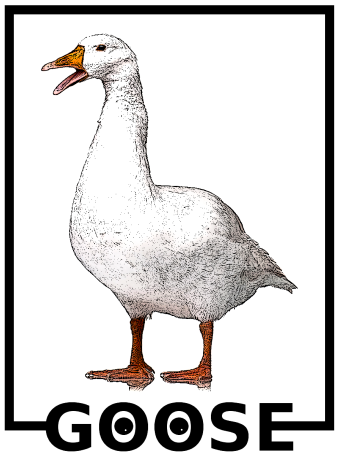

A toolkit for DNA sequence analysis and manipulation

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

1 Introduction **2**

1.1 Installation . . . 2

1.2 License . . . 2

2 FASTQ tools **4**

2.1 Program goose-fastq2fasta . . . 5

2.2 Program goose-fastq2mfasta . . . 6

2.3 Program goose-FastqExcludeN . . . 8

2.4 Program goose-FastqExtractQualityScores . . . 9

2.5 Program goose-FastqInfo . . . 10

2.6 Program goose-FastqMaximumReadSize . . . 11

2.7 Program goose-FastqMinimumReadSize . . . 12

3 FASTA tools **14**

3.1 Program goose-fasta2seq . . . 14

3.2 Program goose-fastaextract . . . 15

3.3 Program goose-fastaextractbyread . . . 16

3.4 Program goose-fastainfo . . . 18

3.5 Program goose-mutatefasta . . . 19

3.6 Program goose-randfastaextrachars . . . 20

4 Genomic sequence tools **23**

5 Amino acid sequence tools **24**

5.1 Program goose-AminoAcidToGroup . . . 24

5.2 Program goose-ProteinToPseudoDNA . . . 25

6 General purpose tools **28**

6.1 Program goose-reverse . . . 28

Bibliography **29**

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-fastqclustreads`
4. `goose-FastqExcludeN`: it discards the FASTQ reads with the minimum number of "N" symbols.
5. `goose-FastqExtractQualityScores`: it extracts all the quality-scores from FASTQ reads.
6. `goose-FastqInfo`: it analyses the basic informations of FASTQ file format.
7. `goose-FastqMaximumReadSize`: it filters the FASTQ reads with the length higher than the value defined.
8. `goose-FastqMinimumLocalQualityScoreForward`
9. `goose-FastqMinimumLocalQualityScoreReverse`
10. `goose-FastqMinimumQualityScore`
11. `goose-FastqMinimumReadSize`: it filters the FASTQ reads with the length smaller than the value defined.
12. `goose-count`
13. `goose-extractreadbypattern`
14. `goose-fastqpack`
15. `goose-fastqsimulation`
16. `goose-FastqSplit`

17. goose-FastqTrimm
18. goose-fastqunpack
19. goose-filter
20. goose-findnpos
21. goose-genrandomdna
22. goose-getunique
23. goose-info
24. goose-mfmotifcoords
25. goose-mutatefastq
26. goose-newlineonnewx
27. goose-period
28. goose-permuteseqbyblocks
29. goose-randfastqextrachars
30. goose-real2binthreshold
31. goose-reducematrixbythreshold
32. goose-renamehumanheaders
33. goose-searchphash
34. goose-seq2fasta
35. goose-seq2fastq
36. goose-SequenceToGroupSequence
37. goose-splitreads
38. goose-wsearch

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 paramters.

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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>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
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

Output

The output of the `goose-fastq2fasta` program a FASTA file.
An example, for the input, is:

```
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
```

2.2 Program goose-fastq2mfasta

The `goose-fastq2mfasta` onverts a FASTQ file format to a pseudo Multi-FASTA file. However, it does not align the sequence. Also, it extracts the sequence and adds a pseudo header.

For help type:

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

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

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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>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
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

Output

The output of the `goose-fastq2mfasta` program a Multi-FASTA file.
An example, for the input, is:

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

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 paramters.

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 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
GNNTGATGGCCGCTGCCGATGGCGNANAATCCCACCAANATACCCTTAACAACTTAAGGGTTNTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)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:

```
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
NTTCAGGGATACGACGNTTGTATTTTAAGAATCTGNAGCAGAAGTCGATGATAATACGCGNCGTTTTATCAN
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
Total reads    : 2
Filtered reads : 1
```

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 paramters.

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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>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
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGII>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:

```
IIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>IIIIII/
IIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIGII>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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIIIIIII>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
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIGII>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 filtered reads>
Total reads    : value
Filtered reads : value
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

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
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
Total reads    : 2
Filtered reads : 1
```

2.7 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 paramters.

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 filtered reads>
Total reads    : value
Filtered reads : value
```

An example on such an input file is:

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCCTTAACAACTTAAGGG
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=60
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIIDIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
```

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
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGTCGATGATAATACGCGTCGTTTTATCAT
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIIIIGII>IIIII-I)8I
Total reads    : 2
Filtered reads : 1
```

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 substitues in the DNA sequence the outside ACGT chars by random ACGT symbols.

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 paramters.

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
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `goose-fasta2seq` program is a group sequence.
An example, for the input, is:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

3.2 Program goose-fastaextract

The `goose-fastaextract` extracts sequences from a FASTA file, which the range is defined by the user in the parameters.
For help type:

```
./goose-fastaextract -h
```

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

Input parameters

The `goose-fastaextract` program needs two paramenters, which defines the begin and the end of the extraction, and two streams for the computation, namely the input and output standard. The input stream is a FASTA file.
The attribution is given according to:

```
Usage: ./goose-fastaextract [options] [[--] args]
   or: ./goose-fastaextract [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-fastaextract -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
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `goose-fastaextract` 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 paramters.

Input parameters

The `goose-fastaextractbyread` program needs two paramenters, which defines the begin and the end of the extraction, and two streams for the computation, namely the input and output standard. The input stream is a FASTA or Multi-FASTA file.
The attribution is given according to:

```
Usage: ./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
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `goose-fastaextractbyread` program is FASTA or Multi-FASTA file wiht 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
```

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 paramters.

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
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
```

```
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
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 paramenters 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 paramters.

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.fast
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Output

The output of the `goose-mutatefasta` program is a FASTA or Multi-FASTA file whith 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
ACGCAACGNATTCCTGCTGATCATANTGTNCCGCNCCCCNGCGACGGGGNCTCNCNNGCACACATNGTACCATTGTCCAC
NCTTNCANGTNANCGCTAGCAGGCTACNGTTTNTCCTCNCCTANNCCAANCNGGCGTNNNTACACTGGCACGTGCAGGCA
TNGGTCGGCNGGNNCCTCCGGNAACGGCACCGGAGACGAAGCTCGGNGGNTATACAGGTGTCANGAAACATCCCCGCGNC
GNGTGNCCNNGAANCCANAGAGTATCTCACTCACAACCCTGCGTGCACNTCTAGAGNANGACCTTACNCACCNTCCCNTT
NNGTACCACACCAATGAACGCTGCAGAAAGTCTGTTTNNAGGNGNGCA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ATTTGAAGGCAANCGGNCCAGNAATNCGGNGGGTGCNGCTCNTGTNGGCTACGGNCATCGCGGCCCTGCTNTANTAAGCN
TGAACCACCGNTCGNNGCACTTAGCAATNGCGNAANCCGTCGGCACGGCGGAGACNAANCCGCTANTNNTTTCCCGCTNA
ATGGNTGTACAAGACCNACTANACCANCCTCCGTCACCACACTGGAGCGCANGATGGNNCGCTGNCTAGNAGNCNNTGAG
GCGCTCCNTCCTANAAANCCGTGGNCGAGCNCCCTATGGNAGNGTGGGGGTTTTACCGGAAGACCNTCGNGCCCTATGGG
AGCAATCANAANCTAGAAAGCTTACNGATGGTGANGAANTAGACTANG
```

3.6 Program goose-randfastaextrachars

The `goose-randfastaextrachars` substitues 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 substitues 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
ANAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGAGGGTCCNCCGCTGCCCTGCTGCCATTGNCNCC
NGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGGCAGGAA
GNGGTTTGAGTGGACCTCCNGGCCCCTCATAGGAGAGGAAGCNNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNC
GCGAATCCGNGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCNNN
TAAANNNTCACCCATGAATGCTCACGCAANTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGNGCGCCGGGACAGAATCTCCTTCTCCACCCCCCCNNNTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACC
NGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCNGGAAGCGG
ANAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGNCCCTGGCNCCAGGGTCCNCCGCTGCCCTGCTGCCATTGN
GAGGAAGCNNGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGNCGNGGTTTGAGTGGACCTCCNGGCCCCTCATAGGA
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
ATAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCCCCGCTGCCCTGCTGCCATTGTCCCC
TGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGGCAGGAA
GAGGTTTGAGTGGACCTCCCGGCCCCTCATAGGAGAGGAAGCCGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGTC
GCGAATCCGGGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCTTG
TAAAAGATCACCCATGAATGCTCACGCAAATTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
GCGAATCCGTGCGCCGGGACAGAATCTCCTTCTCCACCCCCCCATCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACC
GGCCCCACCTAAGGAAAAGCAGCCTCCAGGAACTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCGGGAAGCGG
AGAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGTCCCTGGCTCCAGGGTCCTCCGCTGCCCTGCTGCCATTGC
```

```
GAGGAAGCGGGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGGCGCGGTTTGAGTGGACCTCCTGGCCCCTCATAGGA
TCACGCAACTTTAATTACAGACCTGAATAAAATGTCACCCATGAATGC
```

Chapter 4

Genomic sequence tools

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

1. `goose-mutatedna`
2. `goose-randseqextrachars`
3. `goose-geco`
4. `goose-gede`

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 paramters.

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:

```
IPFLLKKQFALADKLVLSKLRQLLGGRIKMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPNSIG
TLMPKAEVKIGENNEILVRGGMVMKGYYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

The output of the `goose-AminoAcidToGroup` program is a group sequence.
An example, for the input, is:

```
HSHHHPPUHHHHNPHHHUPHPUHHSSPHPHHSSSSHPHNSHHSHHHPHHSHUHPHSHSHUNUUHUHUSHPNHUHUSUUHS
UHHSPHNHPHSNUUNHHHPSSHHHPSHHPPSNNUHUHHUNNSHHPUSNHSNHNNUSUHHHUNPHPNHHPUUUSPHHHSUH
HNUPHSPNPHHNUHHHHHNHPPHHUHHHHSSHNUHNNHHPUHUHPHPNPHNHHPUUNHHPHHN
```

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 paramters.

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:

```
IPFLLKKQFALADKLVLSKLRQLLGGRIKMMPCGGAKLEPAIGLFFHAIGINIKLGYGMTETTATVSCWHDFQFNPNSIG
TLMPKAEVKIGENNEILVRGGMVMKGYYKKPEETAQAFTEDGFLKTGDAGEFDEQGNLFITDRIKELMKTSNGKYIAPQY
IESKIGKDKFIEQIAIIADAKKYVSALIVPCFDSLEEYAKQLNIKYHDRLELLKNSDILKMFE
```

Output

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

An example, for the input, is:

```
ATCCCGTTTCTGCTGAAAAAACAGTTTGCACTGGCAGACAAACTGGTACTGTCTAAACTGCGTCAGCTGCTGGGCGGCCG
TATCAAAATGATGCCGTGCGGCGGCGCAAAACTGGAGCCGGCAATCGGCCTGTTTTTTCATGCAATCGGCATCAACATCA
AACTGGGCTACGGCATGACGGAGACGACGGCAACGGTATCTTGCTGGCATGACTTTCAGTTTAACCCGAACTCTATCGGC
ACGCTGATGCCGAAAGCAGAGGTAAAAATCGGCGAGAACAACGAGATCCTGGTACGTGGCGGCATGGTAATGAAAGGCTA
CTACAAAAAACCGGAGGAGACGGCACAGGCATTTACGGAGGACGGCTTTCTGAAAACGGGCGACGCAGGCGAGTTTGACG
AGCAGGGCAACCTGTTTATCACGGACCGTATCAAAGAGCTGATGAAAACGTCTAACGGCAAATACATCGCACCGCAGTAC
ATCGAGTCTAAAATCGGCAAAGACAAATTTATCGAGCAGATCGCAATCATCGCAGACGCAAAAAAATACGTATCTGCACT
GATCGTACCGTGCTTTGACTCTCTGGAGGAGTACGCAAAACAGCTGAACATCAAATACCATGACCGTCTGGAGCTGCTGA
AAAACTCTGACATCCTGAAAATGTTTGAG
```

Chapter 6

General purpose tools

1. `goose-comparativemap`
2. `goose-BruteForceString`
3. `goose-char2line`
4. `goose-sum`
5. `goose-min`
6. `goose-minus`
7. `goose-max`
8. `goose-extract`
9. `goose-segment`
10. `goose-reverse`: it reverses the order of a sequence.

6.1 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 paramters.

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:

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAAACAAGATGCCATTGTCCCCCGGCCTCCTGCTG
CTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCA
GGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCG
GGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCCAGCAATCCGCGCGCCGGGAC
AGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAGTT
TAATTACAGACCTGAA
```

Output

The output of the `goose-reverse` program is a group sequence.
An example, for the input, is:

```
AAGTCCAGACATTAATTTGAACGCACTCGTAAGTACCCACTCCAAAATAAACGTCCTCCTCTTCCAGAAGGTCTTCTTCA
AGGACGTCCCGTAAGACAGGGCCGCGCGCCTAACGACCCCCCCCACGCGGAAGGACGGCGGACCGGTGGAGGGCTCGAAGG
AGAGGATACTCCCCGGGCCGTGACCGGACCCTCCAGGTGAGTTTGGTGGTTCGCTCCTTTCAGTCCTCCGACGAAAAGGA
ATAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGCCACCCCGGTGGGAGGTCCCCGTCCCGTCGCCACCGGCACC
GGGGCCTCTCGTCGTCGTCGTCCTCCGGCCCCCTGTTACCGTAGAACAAAGTCCAGACATTAATTTGAACGCACTCGTAA
GTACCCACTCCAAAATCGACCCCCCCACCTCTTCCAGAAGGTCTTCTTCAAGGACGTCCCGAAACGTCCTCTAAGACAGG
GCCGCGCGCCTAAGCGCCGTGACCGGACGAAGGACGGCGGACCGGTGGAGGGCTCGAAGGAGAGGATACTCCCCGGGCCT
CCAGGTGAGTTTGGTGAAGGACGGCGAAGGACGTATACGAGCGACAGAGCCGGGTTCGCTCCTTTCAGTCCTCCGACGAA
AAGGAATCCACCCCGGCCCCTGTTACCGTCGTCCCGTCGCCACCTGGGAGGTCCCGGCACCGGGGCCTCTCGTCGTCGTC
GTCCTCCGGCAGAACA
```