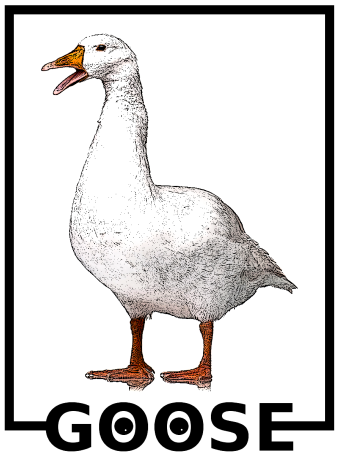


# A toolkit for DNA sequence analysis and manipulation

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

# 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`
2. `goose-fastq2mfasta`
3. `goose-fastqclustreads`
4. `goose-FastqExcludeN`
5. `goose-FastqExtractQualityScores`
6. `goose-FastqInfo`
7. `goose-FastqMaximumReadSize`
8. `goose-FastqMinimumLocalQualityScoreForward`
9. `goose-FastqMinimumLocalQualityScoreReverse`
10. `goose-FastqMinimumQualityScore`
11. `goose-FastqMinimumReadSize`
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

# 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`
6. `goose-randfastaextrachars`
7. `goose-geco`
8. `goose-gede`
9. `goose-mutatedna`
10. `goose-randseqextrachars`
11. `goose-reverse`: it reverses the order of a sequence.
12. `goose-reverselm`: it reverses the order of a large sequence. Low memory usage for large files.

## 3.1 Program goose-fasta2seq

The `goose-fasta2seq` converts a FASTA or Multi-FASTA file format to a seq.
For help type:

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

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

### 3.1.1 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
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

### 3.1.2 Output

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

```
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
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.

### 3.2.1 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
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

### 3.2.2 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.

### 3.3.1 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
```

### 3.3.2 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.

### 3.4.1 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
```

An example on such an input file is:

```
>AB000264 |acc=AB000264|descr=Homo sapiens mRNA
ACAAGACGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCCTGGAGGGTCCACCGCTGCCCTGCTGCCATTGTCCCC
GGCCCCACCTAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTGGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAA
GTGGTTTGAGTGGACCTCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGCAGGCCAGTGCC
```

```
GCGAATCCGCGCGCCGGGACAGAATCTCCTGCAAAGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCACCCCCCCCAGC
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCCCCTGGAGGGT
GGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGCCTCCTGACTTTCCTCGCTTG
GTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAG
GCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCCCTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAA
TAAAACCTCACCCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

### 3.4.2 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.

### 3.5.1 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
```

### 3.5.2 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
```

# Chapter 4

# 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.

## 4.1 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.

### 4.1.1 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
```

### 4.1.2 Output

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

```
HSHHHPPUHHHHNPHHHUPHPUHHSSPHPHHSSSSHPHNSHHSHHHPHHSHUHPHSHSHUNUUHUHUSHPNHUHUSUUHS
UHHSPHNHPHSNUUNHHHPSSHHHPSHHPPSNNUHUHHUNNSHHPUSNHSNHNNUSUHHHUNPHPNHHPUUUSPHHHSUH
HNUPHSPNPHHNUHHHHHNHPPHHUHHHHSSHNUHNNHHPUHUHPHPNPHNHHPUUNHHPHHN
```

## 4.2 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.

### 4.2.1 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
```

### 4.2.2 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 5

# General purpose tools

1. `goose-comparativemap`: visualisation of comparative maps. It builds a image given specific patterns between two sequences.
2. `goose-BruteForceString`: it generates, line by line, multiple combinations of strings up to a certain size.
3. `goose-char2line`: it transforms each char into a char in each line.
4. `goose-sum`: it adds the second column value to the first column value.
5. `goose-min`: it finds the minimum value between two column values.
6. `goose-minus`: it substracts the second column value to the first column value.
7. `goose-max`: it finds the mmaximum value between two column values.
8. `goose-extract`: it extracts a subsequence of a sequence by coordinates.
9. `goose-segment`: it segments a sequence given a certain threshold.