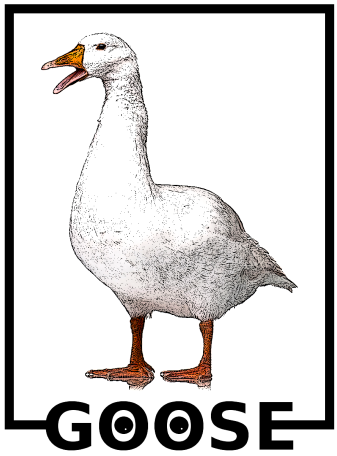

A toolkit for DNA sequence analysis and manipulation

D. Pratas (pratas@ua.pt)
A. J. Pinho (ap@ua.pt)

IEETA/DETI, University of Aveiro, Portugal

Version 1.7.17

Contents

1. Introduction **2**

2. FASTQ tools **3**

3. FASTA tools **5**

4. Sequence tools **6**

4.1 Program goose-AminoAcidToGroup 6

4.1.1 Input parameters 6

4.1.2 Output 7

5. General purpose tools **8**

Note 11: Software availability & characteristics **9**

11.1 FALCON program 9

11.1.1 Input parameters 9

11.1.2 Output 10

11.2 FALCON-FILTER program 10

11.2.1 Input parameters 11

11.2.2 Output 11

Bibliography **11**

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 [1]. Several frameworks for analysis and manipulation emerged, namely GALAXY [2], GATK [3], HTSeq [4], MEGA [5], 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 present GOOSE, a (free) novel toolkit for analyzing and manipulating SEQ ("ACGTN"), FASTA or FASTQ formats, with many complementary tools, 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 high very large datasets, typically GB and TB sizes. For this dimension of data fast access and flexibility is crucial, which are some of the qualities of GOOSE.

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 freely available, under GPLv3, at `https://pratas.github.io/goose`.

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-extract**
14. **goose-extractreadbypattern**
15. **goose-fastqpack**
16. **goose-fastqsimulation**
17. **goose-FastqSplit**
18. **goose-FastqTrimm**
19. **goose-fastqunpack**
20. **goose-filter**

21. **goose-findnpos**
22. **goose-genrandomdna**
23. **goose-getunique**
24. **goose-info**
25. **goose-mfmotifcoords**
26. **goose-mutatedna**
27. **goose-mutatefasta**
28. **goose-mutatefastq**
29. **goose-newlineonnewx**
30. **goose-period**
31. **goose-permuteseqbyblocks**
32. **goose-randfastaextrachars**
33. **goose-randfastqextrachars**
34. **goose-randseqextrachars**
35. **goose-real2binthreshold**
36. **goose-reducematrixbythreshold**
37. **goose-renamehumanheaders**
38. **goose-reverse**
39. **goose-reverselm**
40. **goose-searchphash**
41. **goose-segment**
42. **goose-seq2fasta**
43. **goose-seq2fastq**
44. **goose-SequenceToGroupSequence**
45. **goose-splitreads**
46. **goose-wsearch**

3. FASTA tools

1. **goose-fasta2seq**
2. **goose-fastaextract**
3. **goose-fastainfo**

4. Sequence tools

Current available sequence tools for analysis and manipulation include:

1. **goose-AminoAcidToGroup**
2. **goose-ProteinToPseudoDNA**
3. **goose-geco**
4. **goose-gede**

4.1 Program goose-AminoAcidToGroup

The goose-AminoAcidToGroup converts an amino acid sequence to a group sequence. 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 < in.prot > out.group
It converts a amino acid sequence to a group sequence.
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.

Output

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

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.

Note 11: Software availability & characteristics

All source code for the FALCON is available at:

- `https://github.com/pratas/falcon`.

FALCON includes the following programs (See Note 2 for a common pipeline using the programs):

1. **FALCON** - infers metagenomic composition;
2. **FALCON-FILTER** - filters local similarity of inferred sequences;
3. **FALCON-EYE** - displays the output from FALCON and FALCON-FILTER in a compact map;

FALCON program

The FALCON program enables to identify and quantify the similarity between present-days reference sequences and the FASTQ reads from ancient genomes. In the following subsections, we explain the input and output paramters.

Input parameters

The FALCON program needs two files for the computation. The FILE1, with the reads provided from the NGS platform, and the FILE2, with the multi-FASTA cointaining the sequences of the genomes and respective headers, are mandatory arguments. The rest of the parameters are non mandatory, namely:

```
Usage: FALCON [OPTION]... [FILE1] [FILE2]
A compression-based method to infer metagenomic sample composition.

Non-mandatory arguments:

  -h                        give this help,
  -F                        force mode (overwrites top file),
  -V                        display version number,
  -v                        verbose mode (more information),
  -Z                        database local similarity,
  -s                        show compression levels,
```

```
  -l <level>               compression level [1;44],
  -p <sample>              subsampling (default: 1),
  -t <top>                 top of similarity (default: 20),
  -n <nThreads>            number of threads (default: 2),
  -x <FILE>                similarity top filename,
  -y <FILE>                local similarities filename,

Mandatory arguments:

  [FILE1]                  metagenomic filename (FASTA or FASTQ),
  [FILE2]                  database filename (FASTA or Multi-FASTA).
```

There are two hidden parameters. One for setting the cache size:

```
  -c <cache>             maximum collisions for hash cache. Memory
                         values are higly dependent of the parameter
                         specification.
```

This is a parameter that affects the precision and computational resources needed for the computation. Lower values use less memory. For a complete description see [6, 7]. The previous parameter is only needed when the context of the Markov model is higher than 13. For setting specifically the models through the command line use the following explanation (more information can be found at [6, 7]):

Output

The output of the FALCON program are two files. One file contains the top-n values with the highest similarity relatively to the reads. The following output shows an example of a top-5:

```
1   9472    92.597  548558394   Human endogenous retrovirus K113
2   4668621 42.184  1069460419  Escherichia coli strain 210221272
3   4558287 42.016  1008930592  Shigella sp. PAMC 28760
4   4574246 36.661  844762407   Shigella boydii strain ATCC 9210
5   4878853 36.531  992379426   Shigella sonnei strain FDAARGOS 90
```

These column values stand, respectively, for the ranking of higher similarity, the sequence size, the percentage of similarity, the global identifyer (GI) and the name of the FASTA read (genome name).

The other file contains the relative local similarities and it is only available when FALCON runs with the "-Z" flag. This file is the input of the FALCON-FILTER program that we describe next. This file is packed in a compact format for not increasing the substantially the storage.

FALCON-FILTER program

The FALCON-FILTER program enables to identify and quantify where the similarity between present-days reference sequences and the FASTQ reads from ancient genomes, below a certain threshold, occurs. In the following subsections, we explain the input and output paramters.

Input parameters

The FALCON-FILTER program needs only one file for the computation. The file is provided from the output of the FALCON program (when the flag "-Z" is set on the FALCON program). The non-mandatory arguments are mostly filters and the threshold.

```
Usage: FALCON-FILTER [OPTION]... [FILE]
Filter and segment FALCON output.

Non-mandatory arguments:

  -h                        give this help,
  -F                        force mode (overwrites top file),
  -V                        display version number,
  -v                        verbose mode (more information),
  -s  <size>                filter window size,
  -w  <type>                filter window type,
  -x  <sampling>            filter window sampling,
  -sl <lower>               similarity lower bound,
  -su <upper>               similarity upper bound,
  -dl <lower>               size lower bound,
  -du <upper>               size upper bound,
  -t  <threshold>           threshold [0;2.0],
  -o  <FILE>                output filename,

Mandatory arguments:

  [FILE]                    profile filename (from FALCON).
```

Output

The output of the file contains the coordinates of the relative similar regions with the respective self-similarity classification.

Bibliography

[1] H. Buermans and J. Den Dunnen, "Next generation sequencing technology: advances and applications," *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, vol. 1842, no. 10, pp. 1932–1941, 2014.

[2] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor *et al.*, "Galaxy: a platform for interactive large-scale genome analysis," *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[3] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. Del Angel, M. A. Rivas, M. Hanna *et al.*, "A framework for variation discovery and genotyping using next-generation dna sequencing data," *Nature genetics*, vol. 43, no. 5, pp. 491–498, 2011.

[4] S. Anders, P. T. Pyl, and W. Huber, "Htseq–a python framework to work with high-throughput sequencing data," *Bioinformatics*, p. btu638, 2014.

[5] S. Kumar, G. Stecher, and K. Tamura, "Mega7: Molecular evolutionary genetics analysis version 7.0 for bigger datasets," *Molecular Biology and Evolution*, p. msw054, 2016.

[6] D. Pratas, A. J. Pinho, and P. J. S. G. Ferreira, "Efficient compression of genomic sequences," in *Proc. of the Data Compression Conf., DCC-2016*, Snowbird, Utah, Mar. 2016, pp. 231–240.

[7] D. Pratas, "Compression and analysis of genomic data," Ph.D. dissertation, University of Aveiro, 2016.