Compareads

Comparing huge metagenomic experiments

v2.1

User's guide – May 2014

Authors:
Nicolas Maillet
Claire Lemaître
Rayan Chikhi
Dominique Lavenier
Pierre Peterlongo
Guillaume Collet

Contact:
pierre.peterlongo@inria.fr

Licence

Publication

Compareads was presented in *RECOMB Comparative Genomics* in 2012 in Niterói, Brazil. The publication can be found in BMC Bioinformatics: http://www.biomedcentral.com/1471-2105/13/S19/S10

Software description

Compareads has been developed to enable the comparison of huge metagenomic datasets (files containing reads from metagenomic experiments). The efficiency of Compareads is due to a compact data structure: the **Bloom filter** (Bloom, 1970). Each read is split in k-mers, which are then stored in a Bloom filter by 4 bits, thus reducing the memory footprint. Moreover, to reduce the disk space used to store the results of a comparison, Compareads only stores a vector of bits (called **bit vector**) that represents the selected reads.

To compare two sets of reads, the Compareads approach consists in three steps:

I) Filter the reads given some parameters.
II) Select reads that are similar in both sets.
III) Write the similar reads on disk.

These three steps are implemented in three programs called filter_reads, compare_reads and extract_reads. We also developed a fourth program called bvop, which allows Boolean operations between bit vectors.

Filter_reads corresponds to the filtering step of Compareads. Each read is filtered on three parameters: its size, its N content and its Shannon entropy index (see below). Filter_reads produces a bit vector that represents the selected reads.

Compare_reads corresponds to the comparison step. Given two read sets A and B, reads from A are indexed in a Bloom filter where reads from B are searched in and vice versa. Thus, compare_reads produces two bit vectors that represents:

- Reads from A similar to at least one read in B.
- Reads from B similar to at least one read in A.

Extract_reads extracts reads from a given file based on a given bit vector, and writes them in an output file.

bvop (bit vector operators) allows to combine bit vectors through the following Boolean operators: AND, OR, NOT, ANDNOT.

This version of Compareads has been refactored and written in C++11. Computation times are equivalent with the original version of Compareads.

Programs

Filter_reads

Filter_reads takes a file containing reads and filters them on three parameters, their length, their contents in N (number of unknown bases), and their Shannon entropy index. The output file contains the bit vector corresponding to selected reads.

Usage:

```
./filter_reads <input_file> [options]
```

Input:

The input file need to be in a well-formed **fasta or fatsq** format, compressed with **gzip or not** (errors often comes from bad formatted files).

Output:

The output file is a bit vector that represents the selected reads in the input file. The size of the bit vector is the number of reads in the input file. The default output file name is the input file name with `.bv` extention. The user may also specify the output file name with –o option.

Options:

- `-l int` : minimal length a read should have to be kept [default=0].
- `-n int` : maximal number of Ns a read should contains to be kept [default=infinite].
- `-e float` : minimal Shannon index a read should have to be kept [default=0].
- `-c string` : the given string will be paste in the header of the output file.
- `-m int` : maximum number of selected reads [default=all].
- `-o string` : the output file name [default=stdout].
- `-h` : prints this help.
- `-v` : prints the version number.

Compare_reads

Compare_reads takes two sets of files containing reads and finds the common reads between the two sets. Two reads are considered similar if they share a given number of identical non-overlapping k-mers. Each file may be associated to a .bv file (bit vector) that represents the previously filtered reads.

Be careful: The sets of reads given in input are supposed to be filtered by filter_reads. No filter is made in compare_reads on the size or the complexity of reads.

Usage:

```
./compare_reads –a <file[,bv]> -b <file[,bv]> [options]
```

Input:

Compare_reads takes two sets (*A* and *B*) of files containing reads.

Input files need to be in a well-formed **fasta or fatsq** format, compressed with **gzip or not** (errors often comes from bad formatted files).

The files of set *A* are declared with the –a flag. The files of set *B* are declared with the –b flag.

Input files may have an associated bit vector. A bit vector associated to a file is declared

Output:

For each input file *a* in *A*, a bit vector is written in an output file named **a_in_B.bv** that corresponds to reads from *a* found in *B*. Similarly, for each input file *b* in *B*, a bit vector is written in an output file named **b_in_A.bv** that corresponds to reads from *b* found in *A*.

Be careful: If two input files have the same basename, one will write on the other because the basename only is used to generate the output file names. This may happen for files having the same name in different directories.

A log file, containing information about the comparison, is also written in file `A_VS_B.txt`.

Options:

- `-k int` : size of k-mers (value of k) [default=33].
- `-t int` : minimal number of shared non overlapping k-mers [default=2].
- `-m in t` : maximum number of reads to read per file [default=all]
- `-l string` : path to write log file [default=./].
- `-o string` : path to write output files [default=./].
- `-h` : prints this help.
- `-v` : prints the version number.

Extract_reads

Extract_reads takes a file containing reads and its associated bit vector, then it outputs the selected reads in an output file in the same format than the input file.

Usage:

```
./extract_reads <input_file> <input_bv> [options]
```

Input:

The input file needs to be in a well-formed **fasta or fatsq** format, compressed with **gzip or not** (errors often comes from bad formatted files).

The `input_bv` is the associated bit vector file. The bit vector size must be exactly the number of reads in the input file.

Output:

Extract_reads outputs reads, from the input file, that are selected in the bit vector. The default ouput is the standard output, use –o option to specify an output file.

Options:

- `-o string` : name of the output file [default=stdout].
- `-h` : prints this help.
- `-v` : prints the version number.

Bvop

Bvop is designed to perform Boolean operations between bit vectors. It takes a bit vector file and an optional operation to perform on a second bit vector file. If no operation specified, it just does nothing. Option –I prints the comment and some statistics about the input file.

Usage:

```
./bvop <input_file.bv> [options]
```

Input:

Input files are bit vector files generated by compare_reads, filter_reads or bvop. A bit vector contains a header with comments, then a line with a # and the size of the vector (number of reads), finally the vector of bits (binary format).

Output:

The output file contains the result of the Boolean operation applied to the input file(s).

Options:

- `-n` : performs **NOT** on the `input_file.bv`.
- `-a <file2.bv>` : performs **AND** between `input_file.bv` and `file2.bv`.
- `-o <file2.bv>` : performs **OR** between `input_file.bv` and `file2.bv`.
- `-d <file2.bv>` : performs **ANDNOT** between `input_file.bv` and `file2.bv`.
- `-p <output.bv>` : print result in file `output.bv` [Default=stdout].
- `-i` : prints information about `input_file.bv`.
- `-h` : prints this help.
- `-v` : prints the version number.