

BiOCamLib

BiOCamLib is the [OCaml](#) foundation upon which a number of the bioinformatics tools I developed are built.

It mostly consists of a library — you'll need to clone this repository if you want to manually compile other programs I've developed, notably [SINPle](#) or [KPop](#). You might also use the library for your own programs, if you are familiar with OCaml and patient enough to read the code.

As a bonus, BiOCamLib comes bundled with a few programs:

- `RC`, which can efficiently compute the reverse complement of (possibly very long) sequences. Each sequence should be input on a separate line — lines are processed one by one and not buffered. I use this program in many of my workflows.
- `Octopus`, which is a high-throughput program to compute the transitive closure of strings. For instance, the input

```
A duh
b C
c f e
duh zz x
b c
```

will result in the output

```
A      duh      zz      x
b      C      c      f      e
```

(tab-separated) once processed by `Octopus`. This is useful to cluster things.

- `Parallel`, which allows you to split and process an input file chunk-wise using the reader/workers/writer model implemented in `BiOCamLib.Tools.Parallel`. You can see it as a demonstration of the capabilities of the library, but I also often use it as a useful tool to solve real-life problems.
- `FASTools`, which is a Swiss-knife tool for the manipulation of FASTA/FASTQ files. It supports all formats (FASTA, single- and paired-end FASTQ, interleaved FASTQ) and a simpler tabular format whereby FASTA/FASTQ records are represented as tab-separated lines. It facilitates format interconversions and other manipulations.

Installing `RC`, `Octopus`, `Parallel`, and `FASTools`

⚠ Note that the only operating systems we officially support are Linux and MacOS. ⚠

OCaml is highly portable and you might be able to manually compile/install everything successfully on other platforms (for instance, Windows) but you will have to do it yourself.

There are several possible ways of installing the software on your machine: through `conda`; by downloading pre-compiled binaries (Linux and MacOS x86_64 only); or manually.

Conda channel

🚧 Coming soon! 🚧

Pre-compiled binaries

You can download pre-compiled binaries for Linux and MacOS x86_64 from our [releases](#).

Manual install

Alternatively, you can install `RC`, `Octopus`, `Parallel`, and `FASTools` manually by cloning and compiling the BiOCamLib sources. You'll need an up-to-date distribution of the OCaml compiler and the [Dune package manager](#) for that. Both can be installed through [OPAM](#), the official OCaml distribution system. Once you have a working OPAM distribution you'll also have a working OCaml compiler, and Dune can be installed with the command

```
$ opam install dune
```

if it is not already present. Make sure that you install OCaml version 4.12 or later.

Then go to the directory into which you have downloaded the latest BiOCamLib sources, and type

```
$ ./BUILD
```

That should generate the executables `RC`, `Octopus`, `Parallel`, and `FASTools`. Copy them to some favourite location in your PATH, for instance `~/local/bin`.

How to use `RC`

`RC` has no command line options. It inputs sequences from standard input and outputs their reverse complement to standard output, one sequence at the time. Hence, `RC` can be conveniently used in a subprocess. For example, the command

```
$ echo GATaCa | RC
```

would produce `TGtAaTC`. Note that non-`[ACGTacgt]` characters are output unmodified, so sequence validation and linting must be performed elsewhere whenever they are necessary.

Command line options for `Octopus`

This is the full list of command line options available for the program `Octopus`. You can visualise the list by typing

```
$ Octopus -h
```

in your terminal. You will see a header containing information about the version:

```
This is the Octopus program (version 0.4)
(c) 2016-2023 Paolo Ribeca, <paolo.ribeca@gmail.com>
```

followed by detailed information. The general form(s) the command can be used is:

```
Octopus [OPTIONS]
```

Miscellaneous

Option	Argument(s)	Effect	Note(s)
<code>-V</code> <code>--version</code>		print version and exit	
<code>-h</code> <code>--help</code>		print syntax and exit	

Command line options for `Parallel`

This is the full list of command line options available for the program `Parallel`. You can visualise the list by typing

```
$ Parallel -h
```

in your terminal. You will see a header containing information about the version:

```
This is the Parallel program (version 0.4)
(c) 2019-2022 Paolo Ribeca, <paolo.ribeca@gmail.com>
```

followed by detailed information. The general form(s) the command can be used is:

```
Parallel [OPTIONS] -- [COMMAND TO PARALLELIZE AND ITS OPTIONS]
```

Command to parallelize

Option	Argument(s)	Effect	Note(s)
<code>--</code>		consider all the subsequent parameters as the command to be executed in parallel. At least one command must be specified	(mandatory)

Input/Output

Option	Argument(s)	Effect	Note(s)
<code>-l</code> <code>--lines-per-block</code>	<code><positive_integer></code>	number of lines to be processed per block	default= <code>10000</code>
<code>-i</code> <code>--input</code>	<code><input_file></code>	name of input file	default= <code>stdin</code>
<code>-o</code> <code>--output</code>	<code><output_file></code>	name of output file	default= <code>stdout</code>

Miscellaneous

Option	Argument(s)	Effect	Note(s)
<code>-t</code> <code>--</code> <code>threads</code>	<code><positive_integer></code>	number of concurrent computing threads to be spawned (default automatically detected from your configuration)	default= <code>nproc</code>
<code>-v</code> <code>--</code> <code>verbose</code>		set verbose execution	default= <code>false</code>
<code>-d</code> <code>--debug</code>		output debugging information	default= <code>false</code>
<code>-h</code> <code>--help</code>		print syntax and exit	

Command line options for FASTools

This is the full list of command line options available for the program `FASTools`. You can visualise the list by typing

```
$ FASTools -h
```

in your terminal. You will see a header containing information about the version:

```
This is the FASTools program (version 0.5)
(c) 2022-2023 Paolo Ribeca, <paolo.ribeca@gmail.com>
```

followed by detailed information. The general form(s) the command can be used is:

```
FASTools [OPTIONS]
```

Working mode. Executed delayed in order of specification, default=`compact`.

Option	Argument(s)	Effect	Note(s)
compact -c --compact		put each FASTA/FASTQ record on one tab-separated line	
expand -e --expand		split each tab-separated line into one or more FASTA/FASTQ records	
match -m --match	<regex>	select matching sequence names in FASTA/FASTQ records or tab-separated lines. For paired-end files, the pair matches when at least one name matches	
revcom -r --revcom		reverse-complement sequences in FASTA/FASTQ records or tab-separated lines	
dropq -d --dropq		drop qualities in FASTA/FASTQ records or tab-separated lines	

Input/Output. Executed delayed in order of specification, default=-F.

Option	Argument(s)	Effect	Note(s)
-f --fasta	<fasta_file_name>	process FASTA input file containing sequences	
-F		process FASTA sequences from standard input	
-s --single-end	<fastq_file_name>	process FASTQ input file containing single-end sequencing reads	
-S		process single-end FASTQ sequencing reads from standard input	
-p --paired-end	<fastq_file_name1> <fastq_file_name2>	process FASTQ input files containing paired-end sequencing reads	
-P		process interleaved FASTQ sequencing reads from standard input	
-t --tabular	<tabular_file_name>	process input file containing FAST[A Q] records as tab-separated lines	
-T		process FAST[A Q] records in tabular form from standard input	
-l --linter	'none' 'DNA' 'dna' 'protein'	sets linter for sequence. All non-base (for DNA) or non-AA (for protein) characters are converted to unknowns	default= <u>none</u>
--linter-keep-lowercase	<bool>	sets whether the linter should keep lowercase DNA/protein characters appearing in sequences rather than capitalise them	default= <u>false</u>
--linter-keep-dashes	<bool>	sets whether the linter should keep dashes appearing in sequences or convert them to unknowns	default= <u>false</u>
-o --output	<output_file_name>	set the name of the output file. Files are kept open, and it is possible to switch between them by repeatedly using this option. Use '/dev/stdout' for standard output	default= <u>/dev/stdout</u>
-O --paired-end-output	<output_file_name_1> <output_file_name_2>	set the names of paired-end FASTQ output files. Files are kept open, and it is possible to switch between them by repeatedly using this option. Use '/dev/stdout' for standard output	default= <u>/dev/stdout</u> <u>/dev/stdout</u>
--flush --flush-output		flush output after each record (global option)	default= <u>do not flush</u>

Miscellaneous

Option	Argument(s)	Effect	Note(s)
-v --verbose		set verbose execution (global option)	default= <u>false</u>
-h --help		print syntax and exit	