illumiprocessor Documentation

Release 2.0.2

faircloth-lab (Brant C. Faircloth)

Jun 06, 2018

Contents

Release v2.0. (*Changelog*)

illumiprocessor is a tool to batch process illumina sequencing reads using the excellent trimmomatic package. The program takes as input the location of your demultiplexed illumina reads, a configuration file that is formatted in Microsoft Windows INI file format (key:value pairs, see the example file), and the output directory in which you want to store the trimmed reads.

illumiprocessor will trim adapter contamination and low quality bases from SE and PE illumina reads, and the program is also capable of dealing with double-indexed reads . The current version of illumiprocessor uses trimmomatic instead of scythe and sickle (used in v1.x) because we have found the performance of trimmomatic to be better, particularly when dealing with double-indexed illumina reads. However, you may find that running scythe after trimming with illumiprocessor may ensure that every bit of potential adapter contamination is removed.

illumiprocessor is suited to parallel processing in which each set of illumina reads are processed on a separate (physical) compute core. illumiprocessor **assumes that all fastq files input to the program represent individuals samples (i.e., that you have merged mulitple files for each read from the same sample by combining fastq.gz files)**.

CHAPTER 1

Guide

1.1 Purpose

Illumina sequencing offers incredible capacity. Often, biologists wish to split this capacity among many samples run in multiplex. When you do this, and you demultiplex the samples that you run together, you generally get files output from Casava that look something like:

```
acutotyphlops-kunuaensis_CCGGTGG_L005_R1_001.fastq.gz
acutotyphlops-kunuaensis_CCGGTGG_L005_R2_001.fastq.gz

amphisbaena-fuliginosa_GCTCCTC_L005_R1_001.fastq.gz
amphisbaena-fuliginosa_GCTCCTC_L005_R2_001.fastq.gz

boiga-irregularis_GAGAGTA_L005_R1_001.fastq.gz
boiga-irregularis_GAGAGTA_L005_R2_001.fastq.gz
```

These names correspond to a general file structure for each sample that looks like:

```
<name>_<sequence-tag>_<lane>_<read>_<read-file>.fastq.gz
```

where the `<name>` is equivalent to the species binomial name used above, the string of letters is equivalent to `<sequence-tag>`, the `<lane>` the samples were run in was Lane 5 (L005), each sample has a `<read>` Read1 (R1) and Read2 (R2) file because this was paired-end (PE) sequencing, and each file has only a single `<read-file>` (001).

When you multiplex many samples into the same lane, you can end up with many many output files (e.g. 100). But, we still need to trim adapter contamination and low quality bases from these files. Adapter contamination occurs when the insert DNA sequence is shorter than the read length, and we sequence part of the adapter during the sequencing process. Low quality bases occur as a function of read length, chemistry, the sequencing platform, etc. Removal of adapter contamination and low quality bases is essential to proper downstream analysis and processing.

The problem is that with so many files, removing adapter contamination and low quality bases is problematic across all of the files. Doing so may require tedious processing by hand (which is bad) or shell scripting, which many people are

not comfortable with. When we trim adapters and remove contamination, we may also want to do things such as bulk-renaming of the sequence files and placing the resulting trimmed data in a static directory structure for downstream processing (like phyluce).

This is why I wrote illumiprocessor - it processess many input files of Illumina data, in parallel, to:

1. rename hundreds of fastq files
2. create a sample-specific, `adapters.fasta` file for adapter trimming
3. trim adapter contamination from input fastq files
4. trim low quality bases from input fastq files

illumiprocessor is a wrapper script around a software package written in JAVA named trimmomatic that runs trimmomatic against many Illumina fastq files in parallel.

In our hands, trimmomatic is the best adapter and quality trimmer we have used, and it is developed and maintained by Björn Usadel's group. trimmomatic outperforms a number of other read trimmers, it is reasonably fast, and it offers a lots of nice trimming options.

1.2 Installation

The instructions below assume you are either:

1. running osx
2. running linux (x86_64)

We do not support any other platforms, although you should be able to install and run illumiprocessor and its dependencies on various flavors of windows.

1.2.1 trimmomatic & illumiprocessor

You can install the illumiprocessor dependencies in one of two ways: (1) using conda and (2) manually. We **strongly suggest** that you use conda. There are several reasons for this, one being that we can manage lots of **types** of packages with conda. Another is that conda manages dependencies of packages **very, very well**.

Attention: Manual installation of illumiprocessor is not supported.

Attention: illumiprocessor is already installed as part of phyluce. You do not need to install anything else after installing phyluce.

First, you need to install anaconda or *miniconda* **with Python 2.7**. Whether you choose *miniconda* or anaconda is up to you, your needs, how much disk space you have, and if you are on a fast/slow connection.

Attention: You can easily install anaconda or *miniconda* in your `$HOME`, although you should be aware that this setup can sometimes cause problems in cluster-computing situations.

Tip: Do I want anaconda or *miniconda*?

The major difference between the two python distributions is that anaconda comes with many, many packages pre-installed, while *miniconda* comes with almost zero packages pre-installed. As such, the beginning anaconda distribution is roughly 200-500 MB in size while the beginning *miniconda* distribution is 15-30 MB in size.

We suggest that you install miniconda.

Tip: What version of *miniconda* or anaconda do I need?

Right now, illumiprocessor **only runs with Python 2.7**. This means that you need to install a version of *miniconda* or anaconda that uses Python 2.7. The easiest way to do this is to choose carefully when you download a particular distribution for your OS (be sure to choose the Python 2.7 version).

---

miniconda

Follow the instructions here for your platform: https://conda.io/docs/user-guide/install/index.html

---

Note: Once you have installed either Miniconda or Anaconda, we will refer to the install as *conda* throughout the remainder of this documentation.

---

anaconda

Follow the instructions here for your platform: http://docs.continuum.io/anaconda/install.html

---

Note: Once you have installed either Miniconda or Anaconda, we will refer to the install as *conda* throughout the remainder of this documentation.

Checking your *$PATH*

Regardless of whether you install *miniconda* or anaconda, you need to check that you've installed the package correctly. To ensure that the correct location for anaconda or *miniconda* are added to your $PATH (this occurs automatically on the $BASH shell), run the following:

```
$ python -V
```

The output should look similar to (*x* will be replaced by a version):

```
Python 2.7.x :: Anaconda x.x.x (x86_64)
```

Notice that the output shows we're using the *Anaconda x.x.x* version of Python. If you do not see the expected output (or something similar), then you likely need to edit your $PATH variable to add anaconda or *miniconda*.

The easiest way to edit your path, if needed is to open `~/.bashrc` with a text editor (if you are using ZSH, this will be `~/.zshrc`) and add, as the last line:

```
export PATH=$HOME/path/to/conda/bin:$PATH
```

where `$HOME/path/to/conda/bin` is the location of anaconda/miniconda on your system (usually `$HOME/anaconda/bin` or `$HOME/miniconda/bin`).

Warning: If you have previously set your `$PYTHONPATH` elsewhere in your configuration, it may cause problems with your anaconda or *miniconda* installation of illumiprocessor. The solution is to remove the offending library (-ies) from your `$PYTHONPATH`.

Add the necessary bioconda repositories to conda

You need to add the location of the **bioconda_** repositories to your conda installation. To do that, you can follow the instructions at the bioconda site or you can simply edit your `~/.condarc` file to look like:

```
channels:
  - defaults
  - conda-forge
  - bioconda
```

Once you do this, you have access to all of the packages installed at **bioconda_** and **conda-forge_**. The order of this file is important - conda will first search in it's default repositories for package, then it will check conda-forge, finally it will check bioconda.

How to install illumiprocessor

You now have two options for installing illumiprocessor. You can install illumiprocessor in what is known as a conda environment, which lets you keep code for different applications separated into different environments. **We suggest this route**.

You can also install all of the illumiprocessor code and dependencies in your default conda environment.

Install illumiprocessor in it's own conda environment

We can install everything that we need for illumiprocessor in it's own environment by running:

```
conda create --name illumiprocessor illumiprocessor
```

This will create an environment named `illumiprocessor`, then download and install everything you need to run illumiprocessor into this *illumiprocessor* conda environment. To use this illumiprocessor environment, you **must** run:

```
source activate illumiprocessor
```

To stop using this illumiprocessor environment, you **must** run:

```
source deactivate
```

Install illumiprocessor in the default conda environment

We can simply install everything that we need in our default conda environment, as well. In some ways, this is easier, but it could be viewed as a less-ideal option in terms of repeatability and separability of functions. To install illumiprocessor in the default environment, after making sure that you have *miniconda* or anaconda in your $PATH, and after adding the bioconda repositories, run:

```
conda install illumiprocessor
```

1.3 Using illumiprocessor

Using illumiprocessor is reasonably simple after the depdencies are installed. It requires two steps:

1. Create a configuration file telling the program what to do
2. Run the program against the configuration file

1.3.1 Creating a configuration file

The configuration file is structured using the standard python configuration format where sections are denoted using brackets (`[section]`), and key-value pairs are placed under each section given paramters (keys) and their values (values). The illumiprocessor configuration file has four sections:

1. The adapter structure section
2. The sequence tag section
3. The map of sequence tags to samples
4. The map of original sample names to new sample names

The entire file

The entire file structure looks like the following, which is explained in detail, below:

```
[adapters]
i7:AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT

[tag sequences]
BFIDT-030:ATGAGGC
BFIDT-003:AATACTT

[tag map]
F09-44_ATGAGGC:BFIDT-030
F09-96_AATACTT:BFIDT-003

[names]
F09-44_ATGAGGC:F09-44
F09-96_AATACTT:F09-96
```

[adapters]

The adapter structure section is headed by:

```
[adapters]
```

and the heading is followed by two sequences that denote the adapter structures:

```
[adapters]
i7:AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT
```

The adapters are labelled using a **required** naming scheme - the first is prefixed with *i7*, which, if the insert DNA strand is oriented 5' to 3', refers to the "right" side adapter and the second is prefixed with *i5*, which, if the insert DNA strand is oriented the same way, refers to the "left" side adapter.

We use an asterisk to denote where, in the adapter structure, the sample- specific index sequence is added. **This will be filled in automatically for each sample and sample-specific adapter**.

Below are several, indexed adapter structures commonly used with Illumina sequencing.

Illumina TruSeq v3 (single-indexed)

The Illumina TruSeq v3 adapters are largely used with "older" PE library preparations for multiplexed samples. The *i7* adapter contains the sequence tag (AKA "barcode" or "index"), and the system is a "single-indexed" system, meaning that only one index is used to identify samples:

```
[adapters]
i7:AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT
```

Illumina TruSeq HT (double-indexed)

Illumina TruSeq HT adapters are "double-indexed" and both the *i7* and *i5* adapters contain the sequence tag (AKA "barcode" or "index"). Because the system uses two indexes, both indexes are used in concert to identify samples, and the config file needs to have two asterisks indicating where these indexes get inserted:

```
[adapters]
i7:GATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT*GTGTAGATCTCGGTGGTCGCCGTATCATT
```

Illumina TruSeq LT (single-indexed)

You can convert he Illumina TruSeq HT system to what is called the TruSeq LT system by using only one of the two indexes **on the i7 adapter**. This makes the TruSeq LT system functionally equivalent to the older TruSeq v3 system, except that the adapter sequences are different:

```
[adapters]
i7:GATCGGAAGAGCACACGTCTGAACTCCAGTCAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCGTATCATT
```

Illumina Nextera (XT)

The Illumina Nextera system is another way of preparing libraries that uses enzymatic shearing by transposase to (1) shear the DNA and (2) integrate adapters to the DNA for subsequent sequencing. The benefits of this system are speed. The Illumina Nextera system uses double-indexing, similar to the Illumina TruSeq HT system, except that the adapter sequences are different. The config file needs to have two asterisks indicating where these indexes get inserted:

```
[adapters]
i7:CTGTCTCTTATACACATCTCCGAGCCCACGAGAC*ATCTCGTATGCCGTCTTCTGCTTG
i5:CTGTCTCTTATACACATCTGACGCTGCCGACGA*GTGTAGATCTCGGTGGTCGCCGTATCATT
```

[tag sequences]

The tag sequence section maps a "tag name" (or set of tag names) onto the actual tag sequences that correspond to each "tag name". In this way, we're able to refer to the "tag name", which is often clearer than using the tag sequence (which is just a combination of A, C, G, & T).

Single-indexed libraries

The format for single-indexed libraries looks like the following:

```
[tag sequences]
BFIDT-030:ATGAGGC
BFIDT-003:AATACTT
```

This means that there are two libraries we're processing, one with the tag `BFIDT-030` and another with the tag `BFIDT-003`. The sequence on the other side of the colon from each tag name will be inserted into the `i7` adapter at the asterisk.

The tag sequences should be input in 5' to 3' orientation.

Dual-indexed libraries

The format for double-indexed libraries is very similar to the above, except that the section generally contains more tags **and the tag names must include the** `i5` **and** `i7` **designations**:

```
[tag sequences]
i7-N701:GCTACGCT
i7-N702:GGACTCCT
i5-N501:TAGATCGC
i5-N502:CTCTCTAT
```

So, here, we've denoted two different sorts of tags that have names prepended with `i7` and `i5`. This means that the `i7` sequences are those prepended with `i7` and the `i5` sequences are those prepended with `i5`.

The tag sequences should be input in 5' to 3' orientation.

[tag map]

The `[tag map]` section is where we denote which `fastq.gz` files output by the sequencer contain which tags. There are two slightly different formats for this section depending on whether libraries are single-indexed or dual-indexed.

Single-indexed libraries

The format for single-indexed libraries looks like the following:

```
[tag map]
morelia-viridis1_GCCTTCA:BFIDT-030
cnemldophorus-sexlineatus1_GGTACGC:BFIDT-003
```

This means that the fastq.file whose name contains (note the wildcard):

```
morelia-viridis1_GCCTTCA_L005_R*_001.fastq.gz
```

likely contains adapter contamination, and the index used to identify those samples is from BFIDT-030. In other words, here, we're mapping the tag BFIDT-030, whose sequence we denoted before, onto the sample whose fastq files are:

```
morelia-viridis1_GCCTTCA_L005_R1_001.fastq.gz
morelia-viridis1_GCCTTCA_L005_R2_001.fastq.gz
```

Note: You should only input the fastq filename up to the end of the sequence tag - the remainder of the file name is filled in using a relatively standard wildcard structure. If this wildcard structure does not fit your samples, you can run `illumiprocessor` using the `--r1-pattern` and `--r2-pattern` arguments.

Dual-indexed libraries

The format for dual-indexed libraries looks like the following:

```
[tag map]
morelia-viridis1_GCCTTCA:i7-N701,i5-N501
cnemidophorus-sexlineatus1_GGTACGC:i7-N702,i5-N502
```

This means that the fastq.file whose name contains (note the wildcard):

```
morelia-viridis1_GCCTTCA_L005_R*_001.fastq.gz
```

likely contains adapter contamination, and the indexes used to identify those samples is the combination of i7-N701 **and** i5-N501. Here, we're mapping both of the tags i7-N701 and i5-N501, whose sequence we denoted before, onto the sample whose fastq files are:

```
morelia-viridis1_GCCTTCA_L005_R1_001.fastq.gz
morelia-viridis1_GCCTTCA_L005_R2_001.fastq.gz
```

Note: You should only input the fastq filename up to the end of the sequence tag - the remainder of the file name is filled in using a relatively standard wildcard structure. If this wildcard structure does not fit your samples, you can run `illumiprocessor` using the `--r1-pattern` and `--r2-pattern` arguments.

[names]

The names section remaps Illumina-specific file names onto something that's genreally more pleasing for the end-user. For instance, we can place the following into the `[names]` section:

```
morelia-viridis1_GCCTTCA:morelia-viridis1
cnemldophorus-sexlineatus1_GGTACGC:cnemidophorus-sexlineatus1
```

which takes the files originally named:

```
morelia-viridis1_GCCTTCA_L005_R1_001.fastq.gz
morelia-viridis1_GCCTTCA_L005_R2_001.fastq.gz
```

and renames them, after trimming adapter contamination and low-quality bases to (see the *The output directory structure and files* section below for more info):

```
morelia-viridis1-READ1.fastq.gz
morelia-viridis1-READ2.fastq.gz
```

1.3.2 Running the program

Once you have setup the configuration file, the program is ready to run. You run the program using the following:

```
illumiprocessor \
    --input <path-to-directory-of-read-files-to-clean> \
    --output <path-to-directory-of-cleaned-reads-to-output> \
    --config <path-to-config-file> \
    --cores 12
```

--input (required)
The PATH to the input data (a folder of fastq.gz files).

--output (required)
The PATH to where you want to store the output data.

--config (required)
The PATH to the config files.

--cores (optional; default = 1)
The number of compute cores to run simultaneously.

--trimmomatic (optional; default = ~/anaconda/bin/trimmomatic.jar)
The PATH to the trimmomatic jar file.

--min-len (optional; default = 40)
The minimum length of trimmed sequences to output.

--no-merge (optional; default = False)
Do not merge singleton output files.

--r1-pattern (optional; default = None)
A regular expression pattern for matching R1 reads.

--r2-pattern (optional; default = None)
A regular expression pattern for matching R2 reads.

--se (optional; default = False)
Trim single-end (SE) reads.

--phred (optional; default = PHRED33)
The quality scoring system used for the read data (PHRED33 or PHRED64).

--log-path (optional; default = ./)
A path to a directory in which to store the log file(s) output.

--verbosity (optional; default = INFO)
The verbosity level to use for log file output.

1.3.3 The output directory structure and files

After running the program, the resulting directory structure at the requested output path will look like:

```
output-folder/
    morelia-viridis1/
        adapters.fasta
        raw-reads/
            [symlink to R1]
            [symlink to R2]
        split-adapter-quality-trimmed/
            morelia-viridis1-READ1.fastq.gz
            morelia-viridis1-READ2.fastq.gz
            morelia-viridis1-READ-singleton.fastq.gz
        stats/
            morelia-viridis1-name-adapter-contam.txt
    cnemidophorus-sexlineatus1/
        adapters.fasta
        raw-reads/
            [symlink to R1]
            [symlink to R2]
        split-adapter-quality-trimmed/
            cnemidophorus-sexlineatus1-READ1.fastq.gz
            cnemidophorus-sexlineatus1-READ2.fastq.gz
            cnemidophorus-sexlineatus1-READ-singleton.fastq.gz
        stats/
            cnemidophorus-sexlineatus1-name-adapter-contam.txt
```

The "singleton" file

The `singleton` file in the `split-adapter-quality-trimmed` directory contains all of the paired reads that lost their "mate" (the other member of the pair) due to trimming. trimmomatic outputs these reads separately, but illumiprocessor combines them together in a single file.

1.4 Disclaimer: Adapter sequences

CHAPTER 2

Project info

2.1 Citing illumiprocessor

If you use illumiprocessor in your work, you can cite the software as follows:

> Faircloth, BC. 2013. illumiprocessor: a trimmomatic wrapper for parallel adapter and quality trimming. http://dx.doi.org/10.6079/J9ILL.

Please be sure also to cite trimmomatic:

> Lohse M, Bolger AM, Nagel A, Fernie AR, Lunn JE, Stitt M, Usadel B. 2012. RobiNA: a user-friendly, integrated software solution for RNA-Seq-based transcriptomics. Nucleic Acids Res. 40(Web Server issue):W622-7. doi:10.1093/nar/gks540
>
> Del Fabbro C, Scalabrin S, Morgante M and Giorgi FM. 2013. An Extensive Evaluation of Read Trimming Effects on Illumina NGS Data Analysis. PLoS ONE 8(12): e85024. doi:10.1371/journal.pone.0085024

2.2 License

2.3 Changelog

2.3.1 2.0.3

- PEP8 cleanup
- added trove classifiers to setup.py
- uploaded to pypi
- fixed small error in doc relative to –r1-pattern and –r2-pattern

2.3.2 2.0.2

- improve docs
- fix call to /usr/bin/env

2.3.3 2.0.1

- add setup.py for installation
- create conda package
- add better documentation (ongoing)

2.3.4 2.0.0

- use trimmomatic in place of sickle and scythe

2.4 Authors

- Brant Faircloth (brant **at** faircloth-lab **dot** org)

2.5 Contributors

- Mike Harvey (LSU)
- Noor White (Smithsonian Institution)

Index

Symbols

I