Data Wrangling and Processing for Genomics (../)

‹ (../02-quality-control/index.html)

› (../04-variant

Trimming and Filtering

❓ Overview

Teaching: 30 min
Exercises: 25 min
Questions

- How can I get rid of sequence data that doesn't meet my quality standards?

Objectives

- Clean FASTQ reads using Trimmomatic.
- Select and set multiple options for command-line bioinformatic tools.
- Write `for` loops with two variables.

Cleaning Reads

In the previous episode, we took a high-level look at the quality of each of our samples using FastQC. We visualized per-base quality graphs showing the distribution of read quality at each base across all reads in a sample and extracted information about which samples fail which quality checks. Some of our samples failed quite a few quality metrics used by FastQC. This doesn't mean, though, that our samples should be thrown out! It's very common to have some quality metrics fail, and this may or may not be a problem for your downstream application. For our variant calling workflow, we will be removing some of the low quality sequences to reduce our false positive rate due to sequencing error.

We will use a program called Trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic) to filter poor quality reads and trim poor quality bases from our samples.

Trimmomatic Options

Trimmomatic has a variety of options to trim your reads. If we run the following command, we can see some of our options.

Bash

```
$ trimmomatic
```

Which will give you the following output:

Output

```
Usage:
       PE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary <statsSummaryFile>] [-quiet] [-
validatePairs] [-basein <inputBase> | <inputFile1> <inputFile2>] [-baseout <outputBase> | <outputFile1P> <outputFile1U> <outputF
ile2P> <outputFile2U>] <trimmer1>...
   or:
       SE [-version] [-threads <threads>] [-phred33|-phred64] [-trimlog <trimLogFile>] [-summary <statsSummaryFile>] [-quiet] <i
nputFile> <outputFile> <trimmer1>...
   or:
       -version
```

This output shows us that we must first specify whether we have paired end (`PE`) or single end (`SE`) reads. Next, we specify what flag we would like to run. For example, you can specify `threads` to indicate the number of processors on your computer that you want Trimmomatic to use. In most cases using multiple threads (processors) can help to run the trimming faster. These flags are not necessary, but they can give you more control over the command. The flags are followed by positional arguments, meaning the order in which you specify them is important. In paired end mode, Trimmomatic expects the two input files, and then the names of the output files. These files are described below. While, in single end mode, Trimmomatic will expect 1 file as input, after which you can enter the optional settings and lastly the name of the output file.

option	meaning
<inputFile1>	Input reads to be trimmed. Typically the file name will contain an `_1` or `_R1` in the name.
<inputFile2>	Input reads to be trimmed. Typically the file name will contain an `_2` or `_R2` in the name.
<outputFile1P>	Output file that contains surviving pairs from the `_1` file.
<outputFile1U>	Output file that contains orphaned reads from the `_1` file.
<outputFile2P>	Output file that contains surviving pairs from the `_2` file.
<outputFile2U>	Output file that contains orphaned reads from the `_2` file.

The last thing trimmomatic expects to see is the trimming parameters:

step	meaning
`ILLUMINACLIP`	Perform adapter removal.
`SLIDINGWINDOW`	Perform sliding window trimming, cutting once the average quality within the window falls below a threshold.
`LEADING`	Cut bases off the start of a read, if below a threshold quality.
`TRAILING`	Cut bases off the end of a read, if below a threshold quality.
`CROP`	Cut the read to a specified length.
`HEADCROP`	Cut the specified number of bases from the start of the read.
`MINLEN`	Drop an entire read if it is below a specified length.
`TOPHRED33`	Convert quality scores to Phred-33.
`TOPHRED64`	Convert quality scores to Phred-64.

We will use only a few of these options and trimming steps in our analysis. It is important to understand the steps you are using to clean your data. For more information about the Trimmomatic arguments and options, see the Trimmomatic manual (http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf).

However, a complete command for Trimmomatic will look something like the command below. This command is an example and will not work, as we do not have the files it refers to:

Bash

```
$ trimmomatic PE -threads 4 SRR_1056_1.fastq SRR_1056_2.fastq  \
              SRR_1056_1.trimmed.fastq SRR_1056_1un.trimmed.fastq \
              SRR_1056_2.trimmed.fastq SRR_1056_2un.trimmed.fastq \
              ILLUMINACLIP:SRR_adapters.fa SLIDINGWINDOW:4:20
```

In this example, we’ve told Trimmomatic:

code	meaning

code	meaning
`PE`	that it will be taking a paired end file as input
`-threads 4`	to use four computing threads to run (this will spead up our run)
`SRR_1056_1.fastq`	the first input file name
`SRR_1056_2.fastq`	the second input file name
`SRR_1056_1.trimmed.fastq`	the output file for surviving pairs from the `_1` file
`SRR_1056_1un.trimmed.fastq`	the output file for orphaned reads from the `_1` file
`SRR_1056_2.trimmed.fastq`	the output file for surviving pairs from the `_2` file
`SRR_1056_2un.trimmed.fastq`	the output file for orphaned reads from the `_2` file
`ILLUMINACLIP:SRR_adapters.fa`	to clip the Illumina adapters from the input file using the adapter sequences listed in `SRR_adapters.fa`
`SLIDINGWINDOW:4:20`	to use a sliding window of size 4 that will remove bases if their phred score is below 20

Multi-line commands

Some of the commands we ran in this lesson are long! When typing a long command into your terminal, you can use the `\` character to separate code chunks onto separate lines. This can make your code more readable.

Running Trimmomatic

Now we will run Trimmomatic on our data. To begin, navigate to your `untrimmed_fastq` data directory:

Bash

```
$ cd ~/dc_workshop/data/untrimmed_fastq
```

We are going to run Trimmomatic on one of our paired-end samples. While using FastQC we saw that Nextera adapters were present in our samples. The adapter sequences came with the installation of trimmomatic, so we will first copy these sequences into our current directory.

Bash

```
$ cp ~/.miniconda3/pkgs/trimmomatic-0.38-0/share/trimmomatic-0.38-0/adapters/NexteraPE-PE.fa .
```

We will also use a sliding window of size 4 that will remove bases if their phred score is below 20 (like in our example above). We will also discard any reads that do not have at least 25 bases remaining after this trimming step. This command will take a few minutes to run.

Bash

```
$ trimmomatic PE SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz \
                SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz \
                SRR2589044_2.trim.fastq.gz SRR2589044_2un.trim.fastq.gz \
                SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
```

Output

```
TrimmomaticPE: Started with arguments:
 SRR2589044_1.fastq.gz SRR2589044_2.fastq.gz SRR2589044_1.trim.fastq.gz SRR2589044_1un.trim.fastq.gz SRR2589044_2.trim.fastq.gz
SRR2589044_2un.trim.fastq.gz SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
Multiple cores found: Using 2 threads
Using PrefixPair: 'AGATGTGTATAAGAGACAG' and 'AGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'TCGTCGGCAGCGTCAGATGTGTATAAGAGACAG'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTCCGAGCCCACGAGAC'
Using Long Clipping Sequence: 'CTGTCTCTTATACACATCTGACGCTGCCGACGA'
ILLUMINACLIP: Using 1 prefix pairs, 4 forward/reverse sequences, 0 forward only sequences, 0 reverse only sequences
Quality encoding detected as phred33
Input Read Pairs: 1107090 Both Surviving: 885220 (79.96%) Forward Only Surviving: 216472 (19.55%) Reverse Only Surviving: 2850
(0.26%) Dropped: 2548 (0.23%)
TrimmomaticPE: Completed successfully
```

Exercise

Use the output from your Trimmomatic command to answer the following questions.

1) What percent of reads did we discard from our sample? 2) What percent of reads did we keep both pairs?

Solution

1) 0.23% 2) 79.96%

You may have noticed that Trimmomatic automatically detected the quality encoding of our sample. It is always a good idea to double-check this or to enter the quality encoding manually.

We can confirm that we have our output files:

Bash

```
$ ls SRR2589044*
```

Output

```
SRR2589044_1.fastq.gz       SRR2589044_1un.trim.fastq.gz  SRR2589044_2.trim.fastq.gz
SRR2589044_1.trim.fastq.gz  SRR2589044_2.fastq.gz         SRR2589044_2un.trim.fastq.gz
```

The output files are also FASTQ files. It should be smaller than our input file, because we've removed reads. We can confirm this:

Bash

```
$ ls SRR2589044* -l -h
```

Output

```
-rw-rw-r-- 1 dcuser dcuser 124M Jul  6 20:22 SRR2589044_1.fastq.gz
-rw-rw-r-- 1 dcuser dcuser  94M Jul  6 22:33 SRR2589044_1.trim.fastq.gz
-rw-rw-r-- 1 dcuser dcuser  18M Jul  6 22:33 SRR2589044_1un.trim.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 128M Jul  6 20:24 SRR2589044_2.fastq.gz
-rw-rw-r-- 1 dcuser dcuser  91M Jul  6 22:33 SRR2589044_2.trim.fastq.gz
-rw-rw-r-- 1 dcuser dcuser 271K Jul  6 22:33 SRR2589044_2un.trim.fastq.gz
```

We've just successfully run Trimmomatic on one of our FASTQ files! However, there is some bad news. Trimmomatic can only operate on one sample at a time and we have more than one sample. The good news is that we can use a `for` loop to iterate through our sample files quickly!

We unzipped one of our files before to work with it, let's compress it again before we run our for loop.

Bash

```
gzip SRR2584863_1.fastq
```

Bash

```
$ for infile in *_1.fastq.gz
> do
>   base=$(basename ${infile} _1.fastq.gz)
>   trimmomatic PE ${infile} ${base}_2.fastq.gz \
>                ${base}_1.trim.fastq.gz ${base}_1un.trim.fastq.gz \
>                ${base}_2.trim.fastq.gz ${base}_2un.trim.fastq.gz \
>                SLIDINGWINDOW:4:20 MINLEN:25 ILLUMINACLIP:NexteraPE-PE.fa:2:40:15
> done
```

Go ahead and run the for loop. It should take a few minutes for Trimmomatic to run for each of our six input files. Once it's done running, take a look at your directory contents. You'll notice that even though we ran Trimmomatic on file `SRR2589044` before running the for loop, there is only one set of files for it. Because we matched the ending `_1.fastq.gz`, we re-ran Trimmomatic on this file, overwriting our first results. That's ok, but it's good to be aware that it happened.

Bash

```
$ ls
```

Output

```
NexteraPE-PE.fa               SRR2584866_1.fastq.gz         SRR2589044_1.trim.fastq.gz
SRR2584863_1.fastq.gz         SRR2584866_1.trim.fastq.gz    SRR2589044_1un.trim.fastq.gz
SRR2584863_1.trim.fastq.gz    SRR2584866_1un.trim.fastq.gz  SRR2589044_2.fastq.gz
SRR2584863_1un.trim.fastq.gz  SRR2584866_2.fastq.gz         SRR2589044_2.trim.fastq.gz
SRR2584863_2.fastq.gz         SRR2584866_2.trim.fastq.gz    SRR2589044_2un.trim.fastq.gz
SRR2584863_2.trim.fastq.gz    SRR2584866_2un.trim.fastq.gz
SRR2584863_2un.trim.fastq.gz  SRR2589044_1.fastq.gz
```

Exercise

We trimmed our fastq files with Nextera adapters, but there are other adapters that are commonly used. What other adapter files came with Trimmomatic?

Solution

Bash

```
$ ls ~/miniconda3/pkgs/trimmomatic-0.38-0/share/trimmomatic-0.38-0/adapters/
```

Output

```
NexteraPE-PE.fa  TruSeq2-SE.fa    TruSeq3-PE.fa
TruSeq2-PE.fa    TruSeq3-PE-2.fa  TruSeq3-SE.fa
```

We've now completed the trimming and filtering steps of our quality control process! Before we move on, let's move our trimmed FASTQ files to a new subdirectory within our `data/` directory.

Bash

```
$ cd ~/dc_workshop/data/untrimmed_fastq
$ mkdir ../trimmed_fastq
$ mv *.trim* ../trimmed_fastq
$ cd ../trimmed_fastq
$ ls
```

Output

```
SRR2584863_1.trim.fastq.gz    SRR2584866_1.trim.fastq.gz    SRR2589044_1.trim.fastq.gz
SRR2584863_1un.trim.fastq.gz  SRR2584866_1un.trim.fastq.gz  SRR2589044_1un.trim.fastq.gz
SRR2584863_2.trim.fastq.gz    SRR2584866_2.trim.fastq.gz    SRR2589044_2.trim.fastq.gz
SRR2584863_2un.trim.fastq.gz  SRR2584866_2un.trim.fastq.gz  SRR2589044_2un.trim.fastq.gz
```

Bonus Exercise (Advanced)

Now that our samples have gone through quality control, they should perform better on the quality tests run by FastQC. Go ahead and re-run FastQC on your trimmed FASTQ files and visualize the HTML files to see whether your per base sequence quality is higher after trimming.

Solution

In your AWS terminal window do:

Bash

```
$ fastqc ~/dc_workshop/data/trimmed_fastq/*.fastq*
```

In a new tab in your terminal do:

Bash

```
$ mkdir ~/Desktop/fastqc_html/trimmed
$ scp dcuser@ec2-34-203-203-131.compute-1.amazonaws.com:~/dc_workshop/data/trimmed_fastq/*.html ~/Desktop/fastqc_html/trimmed
$ open ~/Desktop/fastqc_html/trimmed/*.html
```

Remember to replace everything between the `@` and `:` in your scp command with your AWS instance number.

After trimming and filtering, our overall quality is much higher, we have a distribution of sequence lengths, and more samples pass adapter content. However, quality trimming is not perfect, and some programs are better at removing some sequences than others. Because our sequences still contain 3’ adapters, it could be important to explore other trimming tools like cutadapt (http://cutadapt.readthedocs.io/en/stable/) to remove these, depending on your downstream application. Trimmomatic did pretty well though, and its performance is good enough for our workflow.

Key Points

- The options you set for the command-line tools you use are important!
- Data cleaning is an essential step in a genomics workflow.

‹ (../02-quality-control/index.html)

› (../04-variant

Edit on GitHub (https://github.com/datacarpentry/wrangling-genomics/edit/gh-pages/_episodes/03-trimming.md) / Contributing (https://github.com/datacarpentry/wrangling-genomics/blob/gh-pages/CONTRIBUTING.md) / Source (https://github.com/datacarpentry/wrangling-genomics/) / Cite (https://github.com/datacarpentry/wrangling-genomics/blob/gh-pages/CITATION) / Contact (mailto:team@carpentries.org)

Using The Carpentries theme (https://github.com/carpentries/carpentries-theme/) — Site last built on: 2019-12-06 12:20:04 +0000.