Trimmomatic - GVA2020

Overview

As mentioned in the introduction tutorial as well as the read processing tutorial, read processing can make a huge impact on downstream work. While cutadapt which was introduced in the read processing tutorial is great for quick evaluation or dealing with a single bad sample, it is not as robust as some other trimmers in particular when it comes to removing sequence that you know shouldn't be present but may exist in odd orientations (such as adapter sequences from the library preparation).

> ⚠ **A note on the adapter file used here**
>
> The adapter file listed here is likely the correct one to use for standard library preps that have been generated in the last few years, but may not be appropriate for all library preps (such as single end sequencing adapters, or nextera based preps). look to both the trimmomatic documentation and your experimental procedures at the bench to figure out if the adapter file is sufficient or if you need to create your own

Learning objectives:

1. Install trimmomatic
2. Set up a small script to work around the annoying java invocation
3. Remove adapter sequences from some plasmids and evaluate effect on read quality, or assembly.

Installing trimmomatic

Trimmomatic's home page can be found at this link which includes links to the paper discussing the program, and a user manual. Trimmomatic is far above average for as far as programs go, most will not have a user manual, may not have been updated since originally published, etc. This is what makes it such a good tool.

Checking for installation

To verify you have placed the .jar file in a location that is already in your path try the following java invocation to pull up the help information.

```
java -jar $HOME/local/bin/trimmomatic-0.39.jar
```

If the above command works, jump down to the section on making a bash script. Otherwise continue with the next section to install the program

Installing using wget

In a new web browser window/tab, navigate to the trimmomatic home page. In the Downloading Trimmomatic section; right click on the 'binary' link for version 0.39 and copy that link address.

Which to choose binary files or uncompiled source code

The binary files will be what you want 100 out of 100 times, likely until you begin working with a specific program that you identify bugs in, submit them to the developers, they actually respond (most programs are not in active development), they try to address them, and begin asking you to try using the compiled version to check different scenarios.

Use the wget command to download the link you just copied to a new folder named src in your $WORK directory.

Using the mkdir command to create a folder named 'src' inside of your $WORK directory

```
mkdir $WORK/src
cd $WORK/src
```

If you already have a src directory, you'll get a very benign error message stating that the folder already exists and thus can not be created.

The wget command is very simple. It has 2 parts: 1. the command 'wget', and 2. the location of the file you want to download.

```
wget http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip
```

You should see a download bar showing you the file has begun downloading, when complete the **`ls`** command will show you a new compressed file named "Trimmomatic-0.39.zip". Next we need to uncompress this file, and copy the executable file to a location already in our $PATH variable.

```
unzip Trimmomatic-0.39.zip
cp Trimmomatic-0.39/trimmomatic-0.39.jar $HOME/local/bin
```

If you don't see the zip file or are unable to cd into the 0.39 directory after unzipping it let the instructor know.

To verify you have placed the .jar file in a location that is already in your path try the following java invocation to pull up the help information.

```
java -jar $HOME/local/bin/trimmomatic-0.39.jar
```

When you compare how wordy and complicated that is to the other programs you have encountered in the course, it makes sense that we would want a simpler way of accessing the program which is exactly what we will do next.

Creating bash script to launch trimmomatic

The goal here will be to create an executable file with 2 lines. The first line will specify that it is a bash script. The second line will be the command we want to run, followed by information needed by bash to know to put the rest of the command you type after the executable file name.

1. Launch nano with the command: nano $HOME/local/bin/trimmomatic
2. In the nano window enter the following 2 lines exactly as they are typed

```
#!/bin/bash
java -jar $HOME/local/bin/trimmomatic-0.39.jar $@
```

3. write and exit nano with control+o and control+x
4. make the trimmomatic file you just created executable

making file executable

```
chmod +x $HOME/local/bin/trimmomatic
```

Verify that you have successfully made your bash script by typing '**`trimmomatic`**' (try using the tab key in the middle as further sign things are working) and then press enter. You should see the same help that you saw above.

Trimming adapter sequences

Example generic command

Example command for trimming illumina paired end adapters

```
trimmomatic PE <READ1> <READ2> -baseout Trim_Reads/<basename> ILLUMINACLIP:<FASTAadapterFILE>:4:30:10 MINLEN:30
```

Breaking down the parts of the above command:

Part	Purpose	replace with/note
trimmomatic	tell the computer you are using the bash script/command you just made	
PE	tell trimmomatic program you are using the paired end mode	
<READ1>	fastq file read1 you are trying to trim	actual name of fastq file
<READ2>	fastq file read2 you are trying to trim	actual name of paired fastq file
-baseout	add a prefix to the resulting trimmed files	
Trimreads /<basename>	put all trimmed reads in a new folder. This is very good practice	typically you will replace with the first part of the fastq file name (before the Snumber or Lnumber depending on the file usually)
`ILLUMINACLIP`	tell trimmomatic program how you want to trim the adapters	
`<FASTAadapterFILE>`	name of file you containing your adapters	good practice to copy the fasta file you want to use to the current directory
`:4:30:10`	allow 4 mismatches require 30 bp of overlap between R1 and R2 to identify the fragment as being less than the read length Require 10bp of sequence to match before removing anything	
`MINLEN:30`	discard any reads that are less than 30bp after trimming	

All of the above has been put together using the trimmomatic manual and experience in our lab given the adapters we use and the library kits we prepare the samples with.

What does this look like in practice you may ask? Read on for some examples of trimming a single sample, or trimming a large number of samples.

Trimming a single sample

Get some data

set up directories and copy files

```
mkdir -p $SCRATCH/GVA_trimmomatic_1sample/Trim_Reads
cd $SCRATCH/GVA_trimmomatic_1sample
cp $BI/gva_course/plasmid_qc/E1-7* .

cp $WORK/src/Trimmomatic-0.39/adapters/TruSeq3-PE-2.fa .
```

The ls command should show you 2 gzipped fastq files and a fasta file

Trim the fastq files

The following command can be run on the head node. Like with FastQC if we are dealing with less than say 1-2Million reads, it is reasonable to run the command on the head node unless we have 100s of samples in which case submitting to the queue will be faster as the files can be trimmed all at once rather than 1 at a time. Use what you have learned in the class to determine if you think this command should be run on the head node. (this was covered in more detail in the first part of the evaluating and processing read quality tutorial.)

Figuring out how many reads are in each file

```
zgrep -c "^+$" *.fastq.gz
```

Example command for trimming illumina paired end adapters

```
trimmomatic PE E1-7_S187_L001_R1_001.fastq.gz E1-7_S187_L001_R2_001.fastq.gz -baseout Trim_Reads/E1-7.fastq.gz
ILLUMINACLIP:TruSeq3-PE-2.fa:4:30:10 MINLEN:30
```

Evaluating the output

The last 2 lines of text you get should read:

```
Input Read Pairs: 6891 Both Surviving: 2391 (34.70%) Forward Only Surviving: 1844 (26.76%) Reverse Only
Surviving: 2644 (38.37%) Dropped: 12 (0.17%)
TrimmomaticPE: Completed successfully
```

From the last line we know things worked correctly.

2nd to last line tells us:

1. we had 6891 total reads
2. 34.7% of reads both R1 and R2 were long enough to be kept after trimming
3. 26.76% of reads and 38.37% of reads only 1 of the reads were long enough and/or not a complete duplicate of the other read
4. only 0.17% of reads were discarded for both R1 and R2. This is a rough estimate of adapter dimer being present in the sample.

When we interrogate the Trim_Reads folder with ls we see 4 files:

1. _1P
2. _2P
3. _1U
4. _2U

1/2 represent read 1 and read2 ... P/U represent paired and unpaired reads. They are kept separate as many programs require R1 and R2 files to be the same length when being used as input.

Trim all the samples from the multiqc tutorial

As mentioned above, if you have already done the multiqc tutorial, you can use your new trimmomatic command to remove the adatper sequences from all 544 samples.

Get some data

set up directories and copy files

```
mkdir -p $SCRATCH/GVA_trimmomatic_multiqcSamples/Trim_Reads
cd $SCRATCH/GVA_trimmomatic_multiqcSamples
cp -r $BI/gva_course/plasmid_qc/ Raw_Reads/
cp $WORK/src/Trimmomatic-0.39/adapters/TruSeq3-PE-2.fa .
```

The ls command will now show 2 directories, and a fasta file.

Trim the fastq files

Just as we used a for loop to set up a set of FastQC commands in the multiqc tutorial, we can use a similar for loop to generate a single file with 272 trim commands for the 544 total files.

The following command will pair all R1 reads in the Raw_Reads folder with its R2 pair, determine the base name, and generate a command to trim the file

for loop to generate all trim commands needed

```
for r1 in Raw_Reads/*_R1_*.fastq.gz; do r2=$(echo $r1|sed 's/_R1_/_R2_/'); name=$(echo $r1|sed 's/_R1_001.fastq.
gz//'|sed 's/Raw_Reads\///'); echo "trimmomatic PE $r1 $r2 -baseout Trim_Reads/$name.fastq.gz ILLUMINACLIP:
TruSeq3-PE-2.fa:4:30:10 MINLEN:30";done > trim_commands
```

Use the **head** and **`wc -l`** to see what the output is and how much there is of it respectively.

Again as we discussed in the multiqc tutorial, running this number of commands is kind of a boarder line case, there are not a lot of total reads, but there are a large number of samples so we are likely to benefit from not being run on an idev node.

submit the job to run on the que

```
launcher_creator.py -N 1 -w 48 -n "trimmomatic" -t 00:10:00 -a "UT-2015-05-18" -j trim_commands -m "module load
launcher/3.0.3"
sbatch trimmomatic.slurm
```

The job should take less than 5 minutes once it starts, the showq -u command can be used to check for the job finishing.

Evaluating the output

submit the job to run on the que

```
ls Trim_Reads | wc -l
grep -c "TrimmomaticPE: Completed successfully" trimmomatic.e*
```

The above 2 commands are expected to show `1088` and `272` respectively, if you see other answers it suggests that something went wrong with the trimming command itself. If so remember I'm on zoom if you need help looking at whats going on.

Beyond the job finishing successfully, the best way to evaluate this data would actually be to move back to the multiqc tutorial and repeat the analysis there that was done on the raw files for the trimmed files here.

Personal experience

The for loop above focuses just on generating the trim commands. In my experience that is only half of the job, the other half is capturing the individual outputs so you can evaluate how many reads were trimmed in what way for each sample. Perhaps you will find this command helpful in your own work:

command taken from my history to trim all files

```
mkdir trimLogs; mkdir Trim_Reads; for r1 in Raw_Reads/*_R1_*.fastq.gz; do r2=$(echo $r1|sed 's/_R1_/_R2_/');
name=$(echo $r1|sed 's/_R1_001.fastq.gz//'|sed 's/Raw_Reads\///'); echo "trimmomatic PE $r1 $r2 -baseout
Trim_Reads/$name.fastq.gz ILLUMINACLIP:TruSeq3-PE-2.fa:4:30:10 MINLEN:30 >& trimLogs/$name.log"; done >
trim_commands
```

After the job completes, the following command is useful for evaluating its success:

command taken from my history to trim all files

```
echo -e "\nTotal read pairs:";wc -l trim_commands;echo "Successful trimmings:"; tail -n 1 trimLogs/*|grep -c
"TrimmomaticPE: Completed successfully"
```

This gives me a quick readout of if all of the individual commands finished correctly.

Further, if you were to use this for loop rather than the one listed in the tutorial above, you would see each sample generates its own log file in the the trimLogs directory that when investigated with cat/more/less/tail/nano is identical to the output you saw when you trimmed a single sentence allowing you to figure out how different samples are being processed.

Optional next steps:

1. Consider moving back over to the multiqc tutorial use multiqc to determine well the trimming worked. \
2. The reads could then further be assembled in the Spades tutorial as they are all plasmid samples.

Return to GVA2020