VA_trimmer user manual

Zhigang Wu [*1] and Renyi Liu [†1]

[1]Department of Botany and Plant Sciences, University of California Riverside

October 15, 2012

1 Introduction

Adaptor trimming is an essential task before doing any serious analysis. Most of current available adaptor trimming tools, which do not allow mismatches and indels. However, sequencing error is unavoidable in most, if not all, of sequencing platforms. Using these adaptor trimming tools will leave adaptors carrying sequencing error untrimmed. This undoubtedly will cause artificially and non-biologically meaningful bias, which is undesirable. Additionally, current adaptor trimming tools seems also having trouble to handle sequences having ambiguous IUPAC nucleotides(one position have several alternative nucleotides). They are very widely used in studies aims to investigate microbial community diversity. If ambiguous IUPAC nucleotide only occur in single position, one can circumvent the limit of not allowing mismatch and indel exerted by current adaptor trimming tools by running the adaptor trimming tools couple times with different combinations of adaptors. However, if ambiguous IUPAC nucleotides occurs in multiple positions, such as the universal IUPAC primer used to amply Archea 16S contains 5 IUPAC nucleotide position (GYGCASCAGKCGMGAAW which is equivalent to G[CT]GCA[CG]CAG[GT]CG[CA]GAA[AT]). It would be extremely cumbersome to trim (at least 2^5 combination). We need a versatile adaptor trimming tool that is able to handling sequencing error and ambiguous IUPAC nucleotides to cope with the rapid advancement of sequencing technology. VA_trimmer is a trimming tool implemented in C++ that handles these two problems efficiently. It has three modes: 1) the dynamic programming mode, which is based on dynamic programming, handles sequence contains mismatches and indels efficiently; 2) the IUPAC mode, which is based on regular expression search engine, processes ambiguous IUPAC adaptor efficiently; in the other mode one can simply cut any specified number of heading and tailing bases. Generally, the tool is very straightforward to use. We have carefully chosen the default value for all optional options so that it requires minimal effort from the user. But also we tried to give the user the flexibility of customizing the tool to tail their need by making these options available. For example, it takes both FASTA and FASTQ formats and you can use VA_trimmer as a typical UNIX command tool which can take I/O by piping. Most importantly, VA_trimmer is fast and competes most available tools.

[*]zhigang.wu@email.ucr.edu

[†]renyi.liu@ucr.edu

2 Main features

1. allowing mismatch
2. supporting IUPAC adaptor trimming
3. supporting case-insensitive trimming
4. supporting cut leading and tailing bases
5. supporting simple regular expression trimming by using []
6. auto format detection for FASTA/Q

3 Sample usages

3.1 Test VA_trimmer dynamic programming mode.

- Here we instruct the program to take input from arbitrary number of FASTQs using pipe, cut both 5' and 3' adaptors and force it to use exact match for both 5' adaptor (via -l option) and 3' adaptor (via -r option). We also instruct the program to write sequences with adaptor being found to with_5_adaptor and sequences with no adaptor being found to no_5_adaptor, both of which will be write to STDOUT by default. For illustration purpose, here I am catting from two same FASTQs, you can of course cat from different FASTQs in reality.

```
$ cat data/adaptor_test_data.fastq data/adaptor_test_data.fastq \
| ./VA_trimmer -I -o with_5_adaptor -n no_5_adaptor \
-5 IamasINGLEADAPT -3 IAMARiGHTADAPTOR -l 0 -r 0
```

- Same as above but taking input from an arbitrary number of files (-i) and trimming adaptors using default parameters (editing distance), which is 20**This is the parameter most of user should use.** Don't worry about we are setting the edit distance too high here, if there are several alignments between adaptor and sequence meeting the requirement, program will always **ONLY** report the best alignment.

```
$ ./VA_trimmer -I -o with_5_adaptor -n no_5_adaptor \
-i data/adaptor_test_data.fastq data/adaptor_test_data.fastq \
 -5 IamasINGLEADAPT -3 IAMARiGHTADAPTOR
```

3.2 Test VA_trimmer IUPAC mode.

- Cut the 5' adaptor with case-insensitive (-I) and IUPAC (-U) mode ON.

```
$ cat data/AS10.fastq | ./VA_trimmer -I -5 GYGCASCAGKCGMGAAW \
-o with_5_adaptor -n no_5_adaptor -U
```

- VA_trimmer also support simple regular expression only allowing use of square brackets [] to denote alternative nucleotides.

```
$ cat data/AS10.fastq | ./VA_trimmer -I \
-5 G[CT]GCA[CG]CAG[GT]CG[CA]GAA[AT] \
-o with_5_adaptor -n no_5_adaptor -U
```

3.3 Performance test: comparison of dynamic programming mode and regular expression mode.

- IUPAC mode, which internally using regular expression.

```
$ time ./VA_trimmer data/FS2.fastq -I -5 TGGAGGGCAAGTCTGGTG \
 -o with_5_adaptor -n no_5_adaptor -U

  real 0m0.216s
```

- Dynamic programming mode.

```
$ time ./VA_trimmer data/FS2.fastq -I -5 TGGAGGGCAAGTCTGGTG \
-o with_5_adaptor -n no_5_adaptor -l 0

    real 0m3.347s
```

- CONCLUSION: if you want to use the exact match anyway, then I recommend you using the IUPAC mode because it's 10 times faster than the dynamic program way.

3.4 Test VA_trimmer using leading and tailing bases mode.

- Cut the 5' adaptor with case-insensitive (-I) and IUPAC (-U) mode ON.

```
$ cat data/AS10.fastq | ./VA_trimmer -I -5 GYGCASCAGKCGMGAAW \
-o with_5_adaptor -n no_5_adaptor -U
```

- VA_trimmer also support simple regular expression only allowing use of square brackets [] to denote alternative nucleotides.

```
$ cat data/AS10.fastq | ./VA_trimmer -I \
-5 G[CT]GCA[CG]CAG[GT]CG[CA]GAA[AT] \
-o with_5_adaptor -n no_5_adaptor -U
```

3.5 Demultiplexing.

- One can use VA_trimmer to demultiplexing. As we know,a read is typically composed of three portions: barcode sequence, adaptor sequence and target sequence. We are only interested in target sequence. Here is one example. The file data/multiplexing.fastq contain sequencing reads from different biological treatments. For example, reads contain (ACGCCGCA) are from stress condition, while reads contain ACGTGTTA are from control condition. In all reads, an adaptor with gagtttgatc[atcg]tggctcag **immediately follows the barcode**. So, an intuitive way to get the target sequence is to run the VA_trimmer to pick up the sequence with barcode first then trim the adaptor sequence by run the program again. Below is an

example usage, in which I set the edit distance to be 1 in both steps. You can of course choose your own according to your situation.
Step 1: sort the input into files based on barcodes

```
$ for i in {ATCG, ATCT, ...};
    do ./VA_trimmer -i multiplexed1.fastq multiplexed2.fastq \
   -5 $i -o $i.fastq -l 1 -n no_barcode_found.fastq ;
   done
```

Step 2: trim the adaptor sequence

```
$ for file in {fastq files from previous step};
    do ./VA_trimmer -i multiplexed1.fastq multiplexed2.fastq \
   -5 five_prime_adaptor -o $file.out;
   done
```

- As you can see, above method is inefficient because it iterated the input file two times in order to do barcode trimming and adaptor trimming, respectively. VA_trimmer has been designed to handle multiplexing efficiently by combine together of demultiplexing and 5' adaptor trimming into one step. Below I instruct the program to take input from STDIN by piping, remove two concatenated 5' adaptors (-5) with case-insensitive mode on (-I). Instead of feeding the barcodes and adaptor in two steps, one can directly feed the barcode+adaptor to the program, which are $\underbrace{ACGCCGCA}_{barcode}\underbrace{gagtttgatcntggctcag}_{adaptor}$ and $\underbrace{ACGTGTTA}_{barcode}\underbrace{gagtttgatcntggctcag}_{adaptor}$, respectively. Please note two files: ACGCCGCAgagtttgatcntggctcag with5_adaptor.fastq ACGTGTTAgagtttgatcntggctcagwith_5_adaptor.fastq will be generated in current directories, if you don't specify -o, the result will be directed to STDOUT.

```
$  cat data/multiplexing.fastq | ./VA_trimmer \
   -I -5 ACGCCGCAgagtttgatcntggctcag \
   -5 ACGTGTTAgagtttgatcntggctcag  \
   -o with_5_adaptor
```

4 Companion tools

4.1 Guess_fastq_format

- Guess_fastq_format as its indicates is able to tell the format of FASTQ in terms of fastq-sanger, fastq-solex, fastq-illumina. If your sequence is generated from Roche 454 then its highly possible that your sequence is in fastq-sanger format. You can refer to Nucleic Acids Res. 2010 April; 38(6): 17671771. for more information about FASTQ format. Below is a typical usage of Guess_fastq_format:

```
$ ./Guess_fastq_format data/FS2.fastq
```

4.2 Quality_trimmer

- Given a sequence quality cutoff Quality_trimmer is able to extract the sub-maximum array of sequence positions out. In addition to that, one also need to specify the format of the FASTQ being examined. As mentioned above, you can get this information by running Guess_fastq_format program. Below is a typical usage of Quality_trimmer. I instructed the program to extract the maximum contiguous region with quality score ¿= 20 (-c) from data/AS10.fastq file which is in fastq-sanger format (-f) and write the extracted result which meet the length requirement of 100 (-l) to seqs_nolessthan20.fastq and the sequences failed length filtering to seqs_lessthan20.fastq (-s).

```
$ ./Quality_trimmer -sanger data/AS10.fastq -c 20 \
-l 100 -s seqs_lessthan20.fastq >seqs_nolessthan20.fastq
```

4.3 Using three tools as a whole

```
$ cat data/AS10.fastq | ./VA_trimmer -I -5 GYGCASCAGKCGMGAAW \
 -n no_5_adaptor -U | ./Quality_trimmer -sanger \
 -c 20 -l 100 -s seqs_lessthan20.fastq >seqs_nolessthan20.fastq
```

5 Input/Output format

VA_trimmer can takes both FASTA/Q formats and outputs in the same format as input file.

6 Command options

Commands	*Type*	*Specification*
-h, --help	Boolean	Print this help page. True if present.
-5, --five	str	Five prime adaptor sequence. Any sequence follows this 5' adaptor will be retained and any sequence precedes 5' adaptor (including 5' adaptor) will be trimmed off.
-3, --three	str	Three prime adaptor sequence. Any sequence precedes 3' adaptor will be retained and any sequence follows 3' adaptor (including 3' adaptor) will be trimmed off.
-i,--input	str	Input file, has to be in FASTQ or FASTA format.
-l, --five-mismatch	int	Allowed number of mismatches or gaps between the read sequence and the 5' adaptor sequence. By setting this value to 1, we are allowing two mismatches between adaptor and read sequence or 1 deletion in the read sequence or 2 insertions in the read sequence. So, if you set this value to 2, you multiply by 2 to get a sense of how many of mismatches & indels allowed in the alignment. Since the program guarantees only report the the best alignment, so setting this value the larger the better. Doing so is fine for those sequence with adaptor sequences. However, for those reads without adaptor sequences will be wrongly trimmed. If you set this value, e.g. half of of the adaptor length, for those without adaptors an unpredictable position will be reported. In other words, this way will cause false positive. So, by default this value is set to 20length. In circumstances where you want to do exact match, set this value to 0.
-r, --three-mismatch	int	Generally same as -l option but for 3' adaptor sequence. By default this value is set to 20sequence if you specified a 3' adaptor.
-p, --percent (=0.2)	float	Percent of length of adaptor will be used as parameter for five-mismatch and three-mismatch. For example, if set this number to 0.2, the length of 5' adaptor is the 20 and the length of 3' is 10, then 4 (0.2*20) and 2 (0.2*10) will be will be used as the value for the five-mismatch and three-mismatch, respectively.
-o, --out_with_adaptor	str	Clean sequence with adaptor trimmed will be write to this file. default: STDOUT
-n, --out_no_adaptor	str	Sequence without adaptor being found will be write to this file. default: STDOUT
-a, --out_align, (=Alignment_log_....txt)		If an adaptor is found within a read sequence, the alignment between adaptor(s) and this sequence and the distance between adaptor(s) and ends will be write to this file. The file is generated automatically you don't need to specify anything.
-I, --case-insensitive	Boolean	Toggle to switch case insensitive default is ON. True if present.
-U, --IUPAC	Boolean	Toggle to start IUPAC match, default is OFF. Note if you set IUPAC mode, no mismatch indel will be allowed. This option is incompatible with -l and -r option. True if present.
-H, --head	int	Cut the leading n base from the input seq. If this option is set, it will ignore-l -r -U, -I options which in this case make no sense. The trimmed seq will be print to-o, or STDOUT, those without being trimmed due to will be print to -n or STDOUT.
-t, --tail	int	Cut the tailing n base from the input seq. Others are same as -H option.
--length-cutoff (=0)	int	suppress trimmed sequence length less than this value from output.
-m,--minimum (=3)	int	minimum number of base matches for trimming leading and tailing partial adaptors.