ngsShoRT 2.1 manual

Sari Khaleel (Sari.S.Khaleel.DM AT dartmouth.edu)
Dartmouth Medical School
Last updated 1-13-2014

Table of Contents:

I. Basic trimming concepts applied in ngShoRT:

1. Take a paired-end (PE, forward and reverse read) or single-end files and trim them using a user-specified sequence of trimming methods

2. When trimming reads, it's important to set a minimum read length for trimmed reads. This is particularly useful with some of the de Bruijn graph assemblers (SOAPdenovo and velvet), which discard reads shorter than the K-mer length used for assembly. So, if the K-mer length for your assembly is 21, set min_rl to 21.

3. **A trimmed read is "good"** (and will thus be printed in the final output) if it meets two conditions: its length is >= min_rl **AND** it was not filtered out by the following read-trimming methods: lqr, nperc, ncutoff, 5adpt (kr), qseqB (kr), and qseq0
   - For PE-read pairs, a pair is removed if either or both of its reads are "bad." If only one read is bad, its sister "surviving" read is saved to a surviving_SE_mates.fastq file. If you're assembling your trimmed reads using velvet, you can assemble this file along with the trimmed PE reads:

./velveth output_directory <hash_length> -fastq -shortPaired <shuffled_trimmed_PE_file> -fastq -short /pathj/surviving_SE_mates.fastq

II. Startup tutorial for the impatient:

NOTE: if you copy the commandline off this file without replacing the dashes ("-"), you may get this error: ERROR (main params: methods) : no methods were specified !!

1. **First, Check your CPAN modules:**
   - ngsShoRT requires the perl modules String::Approx and PerlIO::gzip, which can be installed as follows (you'll need admin permissions):

     perl -MCPAN -e shell
     cpan> install String::Approx
     cpan> install PerlIO::gzip

   - See http://www.cpan.org/modules/INSTALL.html for more info on installing the module.

2. **Download and untar ngsShoRT_2.1 in a target directory**
   tar -xvf /path/ngsShoRT_2.1.tar.gz

3. **Run ngsShoRT on the sample_data**

   - Paired-End (PE) fastq files:
     cd <ngsShoRT's path>
     perl ngsShoRT.pl -pe1 sample_data/fastq/SRR065390_1_1st_2000reads.fastq.gz -pe2 sample_data/fastq/SRR065390_2_1st_2000reads.fastq.gz -o sample_data/output_directory -methods 5adpt

     - This trims the gzipped paired end files (pe1 = forward read, pe2 = reverse reads) using the 5adpt (removal of 5' adapters/primers, which by default trims known illumina primers) and prints the output in sample_data/output_directory

     - Your output files should be at sample_data/output_directory:
       - trimmed_SRR065390_1__1st_2000reads.fastq — trimmed pe1 reads
       - trimmed_SRR065390_2__1st_2000reads.fastq — trimmed pe2 reads
       - surviving_SE_mates.fastq — trimmed pe1 or pe2 whose mate read was filtered out during trimming

       - extracted_five_prime_adapter_sequences_at_100_percent_match.txt
       - log.txt
       - final_PE_report.txt — full report of ngsShoRT runtime, % and number of trimmed bases and reads, and method-specific trimming statistics

     - Commonly used options include:
       - -t <number of threads> — default is 10
       - -min_rl <minimum trimmed read length> — default is 21
       - -print_discarded_reads yes — default is no

     - Additional trimming tools can be added to the -methods sequence, e.g., -methods lqr_5adpt will filter out low quality reads before trimming 5'-adapters.
     - We recommend the trimming method sequence -methods lqr_5adpt_tera for filtering low-quality reads (reads with >50% bases having a quality score <2), removing their adapter/primer sequences, and trimming their low-quality 3'-end bases

   - Single-End (SE) fastq file:
     perl ngsShoRT.pl -se sample_data/fastq/SRR065390_1st_2000reads.fastq -o sample_data/output_directory -methods 5adpt

   - QSEQ files:
     - You'll use the same code as the PE files because ngsShoRT can auto-detect fastq and qseq files:

perl ngsShoRT.pl -se sample_data/qseq/SRR065390_1st_2000_reads_qseq.txt -o sample_data/output_directory -methods 5adpt

- However, qseq file quality scoring is based at Phred64 for Illumina 1.8+, so the output files (which will be in fastq format) are going to have this quality scoring as well. If you want them to be Sanger (Phred33) based, add i2s to your method list to convert from illumina to Sanger scoring:

  perl ngsShoRT.pl -se sample_data/qseq/SRR065390_1st_2000_reads_qseq.txt -o sample_data/output_directory -methods 5adpt_i2s

- Working with compressed files:
  - ngsShoRT auto-detects and opens files with the extensions .bz2, .gz, and .zip
  - If you want your trimmed files output to be gzipped, add -gzip to the commandline. For example,

    perl ngsShoRT.pl -se sample_data/qseq/SRR065390_1st_2000_reads_qseq.txt -o sample_data/output_directory -methods 5adpt_i2s -gzip

    Will produce the output file trimmed_SRR065390_1st_2000_reads_qseq.txt.gz

III. Trimming Methods:

Our trimming methods can be divided into quality-trimming methods, which filter low-quality reads (lqr), trim low-quality 3'-ends of reads (TERA), or try to extract a high-quality string from the read (Mott).

Non-quality trimming methods include 3end, 5end, nsplit, nperc, 5adpt, and qseq0. 3end and 5end simply remove a specific number of bases from the 3' and 5' end of reads, respectively. In contrast, nsplit, nperc and 5adpt examine the alien bases (Ns, adapter sequences) in the sequence line to trim reads. Qseq0 is a special case that works only for qseq files. It removes reads whose filtering flag was 0 (i.e., they did not pass filtering during Illumina sequencing analysis).

Non-trimming methods include i2s and s2i, which allow switching the quality scoring of reads from Illumina to Sanger or from Sanger to Illumina, respectively.

1. TERA : (trim by the) Three End Running Average quality score

Illumina reads generally have lower base-call quality towards the 3'-end of reads. Instead of non-discriminately trimming a specific number of bases from the 3'-end of all reads (which can be done using *3end*), without looking at their quality scores, TERA aims to trim low-quality 3'ends

Algorithm: Trim bases from the 3' end of a read, based on the running average quality score, RAQS, of its bases. Starting at the last (the most 3') base of the read, begin counting RAQs of all bases until reaching a base X where RAQS exceeds a cutoff value specified by -tera_avg. If X's 5' index is < min_rl, it's set to min_rl (see above section to understand why we do this). All bases 3' to X-index are trimmed out.

Example: -methods tera -tera_avg 3

2. 5'-adapter trimming (5adpt):

Removal of known (and user-specified) adapter/primer sequences from reads

Algorithm: Trim adapter sequences from reads (in PE reads, this means trim it from the forward and reverse reads). To be more specific, match the adapter sequences to reads, then either remove the matched read or just trim out the matched sequence and all bases 3' to it.

- -5a_f: specifies the adapter sequences file, which can be one of our built-in adapter libraries (illumina or 454, with illumina-genomic as default) or user-specified sequences.
- -fmi : The furthest matching index (i.e., how far into the read should the script be searching for adapter sequences). We recommend setting it to raw_read_length – 10. Read_length obviously depends on your PE file and their read lengths.
- 5a_mp: matching percentage. Default is 100 (which uses regex matching). If 5a_mp is < 100, we use fuzzy matching, implemented by the String::Approx library
- If an adapter sequence is matched, the method will:
    1. Remove the detected adapter sequence and then remove a specific number of bases before and after it (the number of before and after bases is specified in the adapter_sequences file),
    2. Do one of two actions depending on the value of -5a_axn:
        - kr: **K**ill the entire **r**ead
        - ka: **k**ill the detected adapter sequence and all bases **a**fter it

- Available Illumina libraries
    - i-g (Illumina genomic, Default), i-p (Illumina PE), i-m (Illumina multiplex), i-n (Illumina NlaIII), i-d (Illumina DpnII), i-r (Illumina sRNA)

- Available 454 (pyrosequencing) libraries are:
    - p-b (pyroseq basic), p-r (pyroseq sRNA), p-p (pyroseq PE), p-a (pyroseq amplicon)

Example: -methods 5adpt -5a_f i-g -5a_axn ka

- will kill reads that match to an adapter in the illumina-genomic library instead of the default action to just trim the matched sequence and all bases following it (3' to it) in the read

Notes on approximate matching:

- Approximate matching is case-insensitive, and is done according to a user-specified match_percentage. A match_percentage of 90% means that for every 10 bases, only one mismatch is allowed, and so on.

- The measure of approximateness for String::Approx is *Levenshtein edit distance*. More detail on how this String:: Approx works can be found at : http://search.cpan.org/~jhi/String-Approx-3.26/Approx.pm
- Additional approximate matching options are [-5a_ins INT -5a_del INT -5a_sub INT]
  - They refer to the maximum allowed number of insertions, deletions, and substitutions respectively.
  - So, for example (-5a_mp 90 -5a_ins 0 -5a_del 0) means that one mismatched character is allowed in every 10 chars, but it can NOT be a deletion or an insertion. Thus, it can only be a substitution.

3. nsplit

Instead of completely removing a read with N bases (which can be done using our ncutoff and nperc methods), we split the read around the string to save some of the read

Algorithm:

1. Search the read for substrings of consecutive uncalled bases (N, n, .), which we call Nblocks, whose length ≥ min_Nblock_l (min_Nblock_l is a user-specified cutoff). If there are >1 N-strings that satisfy this condition , pick the longer or leftmost one.
2. If the read has such N-block, delete the block and use the bases after and before it to create two new reads, and if this is PE_trimming, pair them with copies of their parent read's sister read.

Example: -methods nsplit -nsplit_len 5

- Will split a read around a string of ≥ 5 Ns if it finds one

4. lqr

Remove low-quality reads from the PE files.

Algorithm:

Given a user-specified Low quality score cutoff (--lqs) and a percentage cutoff for bases whose quality score is <= lqs, which we call lq_p.

- Count the number of bases whose qual score is <=lq. Let's call these LQ bases.
- Label the read "bad" if the percentage of LQ bases. If it's >= LQ_perc_cutoff, "good" otherwise.

Example: -methods lqr -lqs 4 -lqp 50

- Will filter out a read if it has ≥ 50% of bases with a quality score <4

5. mott

Extract the highest-quality string of bases from the read. In other words, trim out low-quality 5' and 3' bases from the read. The Richard-Mott trimming algorithm is described as follows in CLC's manual:

The algorithm:

1. For every base, convert its quality score, Q, to its corresponding Pe (Perror). Pe = 10 ^(-Q/10)
1. So [Q= 0 --> Pe = 1], [Q=2 --> Pe = 0.6], [Q= 10 --> Pe = 0.1], [Q=20 --> Pe = 0.01], [Q=30 --> Pe = 0.001]
2. For every base, calculate its LmP value, which equals (Limit - Pe).
3. For every base (starting from the 3' end for short reads), add its LmP value to a running sum. If the sum drops below zero, set it to zero.
4. When done with the entire sequence, retain the part of the sequence between 1st positive running sum and the highest value of the running sum.

How to choose limit value:

- LmP = mott_limit - Perror[base], where Perror[base] = 10 ^(-Q/10), where Q = quality score
- So, when LmP = 0, mott_limit = 10^(-Q/10)

- At limit = 1 , LmP will be -ve for only bases with Q < 0
- At limit = 0.6, LmP will be -ve for only bases with Q < 2

Example: -methods mott -mott_lim 0.6

- Will basically extract the highest quality substring, with minimum allowed base quality score being 2

6 and 7. ncutoff and nperc

A common trimming approach to to remove all reads with N-bases. However, this may result in removing some reads that contain a small number of N bases and that may still be of use for assembly and mapping

Algorithm: remove reads if the number or percentage of N bases in them exceed a specific cutoff. Defaults are 50 and 50

Example: -methods ncutoff -ncutoff_len 3

- Will filter out reads with ≥ 3 N-bases

8 and 9. 3end and 5end

NGS reads generally have decreasing quality scores (as low as 0) towards their 3'-ends, and removing a specific number of bases from the 3'-end (or 5'-end) of all reads is commonly done to solve this problem. Alternatively, these methods may be used to remove some fixed sequence at the beginning or ends of reads, e.g., a primer

Example: -methods 3end_5end -n3 5 -n5 10

- Will remove 5 bases from the 3'-end of reads and 10 bases from their 5' ends

10 and 11. qseq0 and qseqB (qseq format-specific reads)

- The qseq file format provides a failed_chastity_filter flag that normally marks a read for being filtered from Illumina's qseq output, but there are occasions where this filtering setting is turned off. qseq0 detects this flag in qseq input files and removes any reads still carrying it.
- Earlier versions of Illumina (< 1.8) used a score mapping based at Phred 64 (corresponding to zero quality) and included quality scores 2, 1, and 0, with the 2 score corresponding to the 'B' character, a special indicator for "unknown" quality scores. qseqB was designed to trim reads that contain more 'B'-scored bases than this cutoff.

Algorithm (qseqB): in the global modes, remove a qseq read with ≥ qB_num bases. In local mode, search for strings of ≥ qB_num bases in a read and either trim the entire read (-qB_axn kr) or just the string and all bases 3' to it (-qB_axn ka). Default mode is local, and default action is ka – trim only the string and bases following it.

Example: -methods qseq0_qseqB -qB_mode global -qB_num 10

- Will filter out qseq reads with the 0-chastity_filter flag and also filter reads with ≥ 10 B-scored bases

12. rmHP

Removing homopolymers from reads

Algorithm: search for a homopolymer sequence, h, whose length exceeds a user-specified limit (default is 8) and consists of one of the bases specified by the user (default is all bases – agct)

Example: -methods rmHP -rmHP_ml 10 -rmHP_bases ag

- Will remove only homopolymers of "A" or "G" bases whose length is ≥ 10

13. i2s and s2i

The two most common quality score settings in fastq files are Sanger (where ASCII #33 = zero phred) and old Illumina (pre 1.8+) score with ASCII #64 as the base.

Algorithm: convert scoring from old illumina to Sanger using i2s or from Sanger to old Illumina using s2i

Example: -methods i2s

- This is used for a pre 1.8+ illumina-generated fastq/qseq file (phred 64-based) where the output file will be scored in Sanger (phred 33-based)

IV. Recommended Trimming Methods

- We recommend the trimming methods -methods lqr_5adpt_tera for filtering low-quality reads (reads with >50% bases having a quality score <2), removing their adapter/primer sequences, and trimming their low-quality 3'-end bases

V. Output files

- **For PE input (-pe1 SR_foo_1.fq -pe2 SR_foo_2.fq):**
  - trimmed_<original filename>_foo_1.fastq
  - trimmed_<original filename>_foo_2.fastq
  - surviving_SE_mates.fastq — Contains surviving (widowed) mates. See prev section
  - log.txt — Contains used params and trimmer progress
  - final_PE_report.txt — Contains total and by-method trimming stats

- **For SE input (-se SR_foo.fq):**
  - trimmed_<original filename>_foo.fastq
  - log.txt — Contains used params and trimmer progress
  - final_SE_report.txt — Contains total and by-method trimming stats

VI. ngsShoRT's program structure

- **The single_fQ_read object**
  - Unlike most trimming tools, ngsShoRT does not perform trimming directly on the reads file. Instead, it loads the read's sequence, quality scores, and header into a single_fQ_read object, which comes with its own set of (trimming) methods and properties. For PE reads, there are two single_fQ_read object for the two paired reads, which are then processed as a PE_fQ_pair.
  - The purpose of this somewhat complex method of managing reads is to make the trimming methods and output modules of ngsShoRT format-independent, i.e., ngsShoRT can potentially trim any read format (fastq, fasta.qual, qseq) as long as its components are loaded into a singe_fQ_read object. At the moment, we do that only for fastq and qseq formats.

- **Threading**
  - ngsShoRT's multithreading (using the perl Threads module) implements embarrassingly parallel processing – each thread processes a separate part of the input file, and final trimmed thread outputs are merged in a final processing step.

- **Program architecture**
  - PE files are processed using the process_PE_files.pm module, which splits the files into consecutive sections, each of which is trimmed by a separate thread running the process_PE_files_section.pm module. This module then runs trimming modules (specified by the -methods option) on the reads, producing a trimmed file section (along with any surviving_PE_mates.fastq) files. When all threads are finished trimming their sections, process_PE_files.pm runs a merging module to merge the files into one final trimmed output file.
  - SE files are processed using the process_SE_files.pm module in a similar fashion to the PE files.

VII. References and Suggested Readings

Cox, M. *et al.* (2010) SolexaQA: At-a-glance quality assessment of Illumina second-generation sequencing data. *BMC Bioinformatics*, **11**:485.

CLC bio. CLC Genomics Workbench, User Manual. http://www.clcbio.com/files/usermanuals/CLC_Genomics_Workbench_User_Manual.pdf

FASTX-Toolkit, http://hannonlab.cshl.edu/fastx_toolkit/

Miller,J.R. *et al.* (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, **95**, 315-327.

Schendure,J. and Hanlee,J. (2008) Next-generation DNA sequencing. *Nature biotechnology,* **26**, 1135-1145.

Zerbino,D. and Birney,E. (2008). Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research,* **18** (5):821-829.

Zerbino,D. (2008). Velvet Manual, version 1.1. *Available online at http://www.ebi.ac.uk/~zerbino/velvet/Man ual.pdf*

DiGuistini,S. *et al.* (2009). De novo genome sequence assembly of a filamentous fungus using Sanger, 454 and Illumina sequence data. *Genome* ***Biology,*** 10**:**R94.

Garcia,T.I. *et al.* (2011). Effects of short read quality and quantity on a de novo vertebrate transcriptome assembly. *Comparative Biochemistry and Physiology, Part C. ScienceDirec*t, In Press.

Shulaev,V. *et al.* (2010). The genome of woodland strawberry (*Fragari vesca*). *Nature Genetics,* **43**, 109-116.

Haridas,S. *et al.* (2011). A biologist's guide to de novo genome assembly using next-generation sequence data: A test with fungal genomes. *Journal of Microbiological Methods*, **86**:3, 368-375.

Atherton,R. *et al.* (2010). Whole genome sequencing of enriched chloroplast DNA using the Illumina GAII platform. *Plant Methods*, **6**:22.