
libqcpp Documentation

Release 0.4.0-11-gc6ce70b-dirty

Kevin Murray

May 22, 2017

Contents

| | | |
|----------|--|-----------|
| 1 | libqcpp Applications | 3 |
| 1.1 | Trimit | 3 |
| 2 | Trimit usage tutorial | 5 |
| 2.1 | Quality control on Arabidopsis reads | 5 |
| 3 | libqcpp API | 9 |
| 3.1 | Overview | 9 |
| 3.2 | Streams | 9 |
| 3.3 | Processors | 10 |
| 4 | Developer documentation | 13 |
| 4.1 | Compiling static binaries | 13 |
| 5 | Indices and tables | 15 |

Contents:

Libqcpp ships with some applications built using the library.

Contents

- *libqcpp Applications*
 - *Trimit*
 - * *QC Steps*
 - * *Usage*

Trimit

Trimit ties together several common QC measures applied to short read sequencing data. It works with paired end Illumina and similar sequencing experiments.

QC Steps

- Measure per-base quality scores
- Trim/Merge reads: does a global alignment between read pairs to detect read-through. Read pairs from fragments less than the read length are trimmed at the fragment length, discarding the second read. Read pairs from fragments that are longer than the read length but less than twice the read length are merged. Read pairs from fragments longer than twice the read length are not modified.
- Windowed quality control: a sliding-window based quality score trimmer, which uses a slightly improved version of the [sickle](#) trimming algorithm.
- Optional length filtering and/or truncation

Usage

See *trimit -h*.

Contents

- *Trimit usage tutorial*
 - *Quality control on Arabidopsis reads*
 - * *QC measures*
 - * *QC reports*
 - * *QC-ing the whole sample*

Quality control on Arabidopsis reads

This example is from the [1001 genomes project](#). We will operate on a small set of reads extracted from one sample (SRR1945463).

This tutorial only requires trimit and wget. Trimit can be obtained from [github](#) and then installed (from source of pre-build binaries) according to the [installation instructions on github](#). wget should already be installed on any modern GNU/Linux operating system (`sudo apt-get install wget` on Debian or Ubuntu).

First, we need to download and extract the prepared data.

```
wget -qO - https://github.com/kdmurray91/libqcpp/raw/master/docs/tutorial-data.tar.gz  
↪ | tar xzv
```

The following files should have been created:

- `first-1000.fastq`: The first 1000 read pairs from this sample.
- `has-adaptors.fastq`: A read pair whose insert size is less than the read length, and hence has adaptors in reads.

- `trimmable.fastq`: a read pair with low base quality sequences
- `mergeable.fastq`: a read pair that can be merged

Now, we will QC the first 1000 read pairs using `trimit`

```
trimit first-1000.fastq > first-1000-defaults.fastq
```

That command uses the default values for minimum quality score (25) and minimum read length of reads (1bp, i.e. no filtering). These can both be adjusted:

```
trimit -q 30 -l 50 first-1000.fastq > first-1000-q30l50.fastq
```

QC measures

The files `has-adaptors.fastq`, `trimmable.fastq` and `mergeable.fastq` demonstrate the main modes of operation for `trimit`.

```
# This read pair's insert size is quite small, meaning that there are
# adaptors in the sequences. This results in a single small read containing
# the consensus of the overlapping reads.

trimit has-adaptors.fastq

# This read pair's insert size is larger than the read length, but smaller
# than twice the read length. This means the pair can be merged into a single
# fragment-length read.

trimit mergeable.fastq

# This read pair has reads with 3' ends whose base quality is low, and are therefore
# trimmed from the reads. This results in two shorter reads.

trimit trimable.fastq
```

QC reports

`Trimit` (and `libqc++`) can prepare YAML-formatted reports on QC processing steps.

```
trimit -q 30 -l 50 -y report.yml first-1000.fastq > first-1000-q30l50.fastq

less report.yml
```

This report contains details of all processing steps, including a summary of the number of reads trimmed and merged, and of the per-cycle quality of all reads.

QC-ing the whole sample

If you wish to QC the entire sample these reads come from, please use the following commands.

```
wget -O reads.sra https://sra-download.ncbi.nlm.nih.gov/srapub/SRR1945463

# Dump a fastq file
fastq-dump \
  --split-spot \
  --skip-technical \
  --stdout \
  --readids \
  --define-seq '@$sn/$ri' \
  --define-qual '+' \
  reads.sra > reads.fastq

trimit reads.fastq > reads_qc.fastq

# ALTERNATIVELY, one can pipe the reads directly into trimit:
fastq-dump \
  --split-spot \
  --skip-technical \
  --stdout \
  --readids \
  --define-seq '@$sn/$ri' \
  --define-qual '+' \
  reads.sra \
  | trimit - > reads_qc.fastq
```


Contents

- *libqcpp API*
 - *Overview*
 - *Streams*
 - *Processors*
 - * *AdaptorTrimPE*
 - * *WindowedQCTrim*
 - * *PerBaseQuality*
 - * *ReadLenFilter*
 - * *ReadLenCounter*
 - * *ReadTruncator*

Overview

libqcpp's API is built around two concepts: Stream and Processors.

Streams

`ReadStream` are streams of sequence reads. These streams parse reads from or write reads to a file or stream. `ReadInputStream` and `ReadOutputStream` do so without any manipulation. A `ProcessedReadStream`

processes reads using a pipeline of processors. ThreadedQCProcessor is a high-level, multi-threaded read processor that reads from and writes to files directly. Streams can report, as member variables or as a YAML report, statistics on reads that have been parsed or written.

Processors

Processors mutate or calculate statistics on a read or read pair. They may also report statistics on all reads they have processed in their lifetime, as member variables or as YAML reports.

The following processors are implemented (shown with constructor arguments).

AdaptorTrimPE

```
AdaptorTrimPE(const std::string &name, int min_overlap=10,
               const QualityEncoding &encoding=SangerEncoding);
```

Aligns a read pair to each other, and detect either adaptor read-through, or read overlap. Operates only on paired reads. Yields single ended reads if the read pair is either shorter than the read length (thus each read contains adaptor sequence) or the read ends overlap.

WindowedQCTrim

```
WindowedQualTrim(const std::string &name, int8_t min_quality,
                  size_t min_length, size_t window_size=0,
                  const QualityEncoding &encoding=SangerEncoding);
```

Uses a sliding-window based approach to trim reads at the point where base quality decreases below a threshold. Reads are trimmed at the first position at which a window's mean base quality is below `min_quality`. The 5' end of a read is also trimmed. Reads less than `min_length` bases long are removed from the stream. The window size may be set using `window_size`; a `window_size` value of 0 causes the window length to be 10% of the read length.

PerBaseQuality

```
PerBaseQuality(const std::string &name,
                const QualityEncoding &encoding=SangerEncoding);
```

Records statistics on per-cycle quality across all read sets, reporting the distribution of base quality scores for each cycle.

ReadLenFilter

```
ReadLenFilter(const std::string &name,
               size_t threshold = 1,
               const QualityEncoding &encoding=SangerEncoding);
```

Filters reads less than `threshold` bases out of a stream.

ReadLenCounter

```
ReadLenCounter(const std::string &name,  
               const QualityEncoding &encoding=SangerEncoding);
```

Counts the length distribution of all reads.

ReadTruncator

```
ReadTruncator(const std::string &name,  
              size_t threshold=64,  
              const QualityEncoding &encoding=SangerEncoding);
```

Truncates reads to `threshold` bases long, and removes reads from the stream less than `threshold` bases long.

Developer documentation

Contents

- *Developer documentation*
 - *Compiling static binaries*

Compiling static binaries

```
make -f src/Makefile.static
```


CHAPTER 5

Indices and tables

- `genindex`
- `search`