proovread manual

Thomas Hackl, Frank Förster

Contents

1 Build status

Last build	`https://api.travis-ci.org/BioInf-Wuerzburg/proovread.svg`
Master branch	`https://api.travis-ci.org/BioInf-Wuerzburg/proovread.svg?branch=master`
Develop branch	`https://api.travis-ci.org/BioInf-Wuerzburg/proovread.svg?branch=develo`

The build tests proovread under different Perl versions (5.10.1, 5.12, 5.14, 5.18, 5.20, 5.22) and samtools (1.2, 1.3).

2 What's new

2.1 proovread-2.13.12

- Added support for Travis-CI builds

2.2 proovread-2.12

- revisited **unitig** mode with improved filter for repeat regions
- **dazzling proovread** - experimental support for DALIGNER as unitig mapper (faster/more sensitive than blasr)

2.3 proovread-2.00

- much faster and more sensitive due to **bwa mem** as default mapper
- increased speed and contiguity by using **unitigs** in correction
- optimized for HiSeq & **MiSeq** reads
- compressed BAM intermediates
- efficient threading up to 20 cores and more
- ...

3 Installation

```
# fresh install
git clone --recursive https://github.com/BioInf-Wuerzburg/proovread
cd proovread/util/bwa
```

```
make

# update existing install to latest version
cd proovread/
git pull
git submodule update --recursive
cd /util/bwa
make clean  # important for changes to bwa source code taking effect
make
```

NOTE: proovread comes with its own, modified version of `bwa`. Using it with a standard bwa built will fail.

4 Dependencies

- Perl 5.10.1 or later
  - `Log::Log4perl`
  - `File::Which` (see #17)
- NCBI Blast-2.2.24+ or later
- samtools-1.1 or later

proovread is distributed ready with binaries of SHRiMP2 and BLASR. If you want to employ your own installed version of these mappers, have a look at the `Advanced Configuration` section.

5 Usage

Test your installation by running proovread on the included sample data set.

```
proovread --sample --pre pr-sample
```

Don't run proovread on entire SMRT cells directly, it will only blast your memory and take forever. Split your data in handy chunks of a few Mbp first:

```
# located in /path/to/proovread/bin
SeqChunker -s 20M -o pb-%03d.fq pb-subreads.fq
```

Run proovread on one chunk first.

```
proovread -l pb-001.fq -s reads.fq [-u unitigs.fa] --pre pb-001
```

If things go smoothly, submit the rest. See `Log and statistics` for some notes on how to interpret logging information.

6 Crunching numbers

Just to give you a hands-on idea of what to expect from correction. Here are some stats of the latest correction I ran. It's from one PacBio cell of a 50Mb heterozygous eukaryote genome (I will add some more numbers on other data sets and correction tools soon)

	raw	proovread	lordec
Sequences	56,877	55,493	53,676
Total (bp)	315,511,633	236,687,413	99,379,617
Longest (bp)	27,874	24,682	13,917
Shortest (bp)	1,000	1,000	1,000
N50 (bp)	7,592	6,236	1,877
N90 (bp)	2,887	1,934	1,141

7 Input

7.1 long-reads

Primarily proovread has been designed to correct *PacBio subreads*. You get these reads either from PacBio's SMRT-Portal or by using `dextract` from Gene Myers PacBio assembler DAZZLER, which I would recommend.

In general, reads can be provided in FASTQ or FASTA format. Quality information is used, but only has minor advantages. More valuable are subread information given in default PacBio IDs, which if available are utilized by proovreads `ccseq` module to improve correction performance. Reads shorter then 2x the mean short read length will be ignored.

It is also possible to feed other types of erroneous sequences to proovread, e.g. contigs, 454 reads, ... However, keep in mind that the alignment model for mappings has been optimized for PacBio reads and may produce artifacts in other scenarios. We are currently working on a version optimized for *Oxford Nanopore* data.

7.2 short-reads

For correction of long reads, proovread needs high coverage short read data. Typically these are HiSeq (75-150bp) and MiSeq reads (200-300bp), with overlapping libraries merged (FLASh) for best performance. But also 454 or PacBio CCS reads can be used.

Reads need to have FASTQ/A format and may differ in length. Pairing information are not used. Use of quality trimmed or error corrected reads can improve results.

The recommended coverage for short reads data is around 30-50X and should be specified with `--coverage`. If you have less coverage, it is definitely still worth running proovread. However, it is likely that contiguity will suffer.

Internally, proovread will sample subsets for different iterations, by default 15X for initial runs, 30X for the finishing. For customization of these rates see `sr-coverage` in proovread's config (`Advanced Configuration`).

7.3 unitigs

In addition to short reads, unitigs can/should be used for correction in particular for large data sets (eukaryotes). Unitigs are high-confidence assembly fragments produced by for example ALLPATHS, Meraculous2 or the Celera Assembler. In contrast to contigs, unitigs don't extend past any conflict in the underlying short read data, making them highly reliable.

There are two huge advantages of using pre-computed unitigs:

1. Contiguity: unitigs are longer then corresponding short reads, which makes them easier to align and give better chances to also correct difficult regions.
2. Speed: During unitig computation, all redundancy is removed from the data, creating a minimal set which can be aligned much faster.

However, unitigs only cover regions without conflicts in short read data space. To correct PacBio reads in full length these gaps need to be corrected with primary short read data.

1. dazzling proovread - dazz2sam Currently, support for DAZZLER/DALIGNER is considered experimental. To use dazzler instead of blasr, either export paths or set `daligner-path` and `dazz-db-path` in the config and invoke with modes `sr+dazz-utg / mr+dazz-utg`. In the current implementation, only a single instance of dazzler will be invoked, therefore threading is determined by the thread setup with which daligner has been compiled (default 4).

   Since proovread is designed to operate on BAM/SAM, for the time being, daligner output is internally converted to SAM using a simple parser script (`dazz2sam`). This script also works as a stand-alone tool for dazzler-to-SAM conversion (`proovread/bin/dazz2sam --help`), which might come in handy if one wants to visualize dazzler mappings in common alignment viewers like IGV or tablet.

2. extracting unitigs from ALLPATHS

   ```
   # extract unitigs from allpaths assembly
   allpathslg/bin/Fastb2Fasta IN=reads.unibases.k96 OUT=unitigs.fa
   ```

8 Output

By default, proovread generates six files in the output folder:

.trimmed.f[aq]	high accuracy pacbio reads, trimmed for uncorrected/low quality regions
.untrimmed.fq	complete corrected pacbio reads including un-/ poorly corrected regions
.ignored.tsv	ids of reads and the reason for excluding them from correction
.chim.tsv	annotations of potential chimeric joints clipped during trimming
.parameter.log	the parameter set used for this run

If you are interested in mappings (BAM) and other intermediary files from iterations have a look at `--keep-temporary`.

The phred scores produced by proovread derive from short read support of each base during correction. The values are scaled to realistically mimic sequencing phred accuracies:

Phred	Accuracy	p33
40	99.99	I
30	99.90	?
20	99.00	5
10	90.00	+

9 Log and statistics

proovread generates a comprehensive log on STDERR. The includes fully functional system calls for scripts/tools invoked by proovread. That way, if something goes wrong, its easy to rerun a certain task individually and take a closer look on the issue.

If you want to analyze, how things are going and whether there might be problems with sensitivity etc., the most important information is `Masked: xx%` after each iteration.

```
grep -P 'Running mode|ked :|ning task' proovread.log
[Mon Jan 26 09:52:05 2015] Running mode: blasr-utg
[Mon Jan 26 09:52:51 2015] Running task blasr-utg
[Mon Jan 26 10:00:32 2015] Masked : 55.3%
[Mon Jan 26 10:00:32 2015] Running task bwa-mr-1
[Mon Jan 26 10:21:45 2015] Masked : 76.2%
[Mon Jan 26 10:28:14 2015] Running task bwa-mr-2
[Mon Jan 26 10:37:55 2015] Masked : 92.2%
[Mon Jan 26 10:39:46 2015] Running task bwa-mr-finish
[Mon Jan 26 10:51:19 2015] Masked : 93.0%
```

Masked regions are regions that have already been corrected at high confidence, minus some edge fraction, which remains unmasked in order to serve as seeds for subsequent iterations. After the first iteration, you should have a masking percentage > 50-75%, strongly depending on quality, type and coverage of your data. With each iteration, this value should increase.

Prior to the final iteration, all data is unmasked and the final iteration is run with strict settings on entirely unmasked data. The obtained percentage can be slightly lower as in the last iteration, and is roughly equal to the amount of read bases that will make it to high-confidence .trimmed.fq output.

10 Advanced Configuration

proovread comes with a comprehensive configuration, which allows tuning down to the algorithms core parameters. A custom configuration template can be generated with `--create-cfg`. Instructions on format etc. can be found inside the template file.

11 Hardware and Parallelization

proovread has been designed with low memory node cluster architectures in mind. Peek memory is mainly controlled by the amount of long reads provided. With chunks of less than 20 Mbp it easily runs on a 8 GB RAM machine.

In theory, proovread can be simply parallelized by increasing `--threads`. However, there are single thread steps and other bottlenecks, which at some point render it more efficient, to run e.g. 4 instances at 8 threads in parallel to make full use of a 32 CPU machine.

12 FAQ / General Remarks

1. Why do proovread results from two identical runs differ / Is proovread deterministic?

   One might expect that proovread results are deterministic - meaning reproducible in identical form if input data is identical. This, however, is not the case in a couple of steps:

   a) bwa mem mappings bwa employs heuristics that allow for slightly different results in repeated runs. In particular, one feature is prone to generate differences when employed in proovread's iterative strategy: for performance reasons bwa encodes nucleotides using 2 bits only, meaning bwa only has a four letter alphabet `[ATGC]`. Other bases, including `NNNN` stretches used for masking by proovread, are converted into random `[ATGC]` strings. This, in particular, effects alignments at the margins of masked regions:

```
   orig | ATGAATTGGTTAATCTGC
 masked | ATGAATTGGTNNNNNNNN
   read |    AATTGGTTAAT
        |
rand-01 | ATGAATTGGTAGCCATGG
        |    |||||||
 aln-01 |    AATTGGT
        |
rand-02 | ATGAATTGGTTTATCTGC
        |    |||||||| ||
 aln-02 |    AATTGGTTAAT
```

b) sorting with threshold Whenever there are decisions to make for sorted list in combination with fixed amount of items to keep/remove, things get non-deterministic if identical values in sorting fields occur. In proovread, this for example affects filtering of "best alignments" in bins (localized scoring context).

c) consensus calling 50-50 ratios in base calling will result in one randomly chosen alternative, minimizing a particular bias.

13 Algorithm and Implementation

Algorithm and Implementation are described in detail in the proovread paper.

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

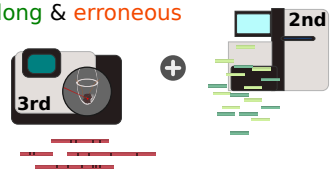
proovread: 3rd generation sequencing length with 2nd generation accuracy

Thomas Hackl[1,2], Felix Bemm[1,2], Frank Förster[2]

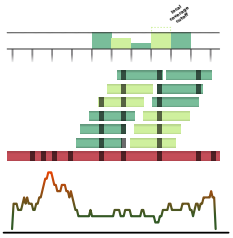
[1] Department of Molecular Plant Physiology and Biophysics, University of Wuerzburg
[2] Department of Bioinformatics, AG Genomics, University of Wuerzburg

short & accurate

long & erroneous

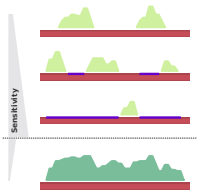


Pacific Bioscience's SMRT sequencing generates exceptionally long reads. But their length comes at the costs of an 15% error rate. Our correction pipeline proovread eliminates these errors in an iterative mapping-consensus approach using high accuracy short read data.



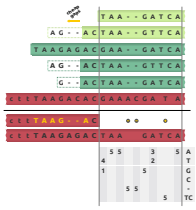
Mapping

Errors in raw single pass PacBio reads are randomly distributed. Common scoring schemes emulate evolutionary sequence changes. We devised a new model for the hybrid alignments reflecting the technical bias. Trusted short read alignments are selected by normalized scores in a local, coverage dependent context to account for the varying error distribution.



Iteration

Sensitive short read mapping on genomic scales is computationally expensive. In our iterative setup, reads are initially mapped at low sensitivity. Regions with sufficient coverage are precorrected and masked. The mapping and correction cycle is restarted with increased sensitivity on masked data. After three iterations, reads are realigned at high specificity. This procedure reduces runtime by more than ten fold compared to a single high sensitivity run.



Consensus

The gap favoring scoring model can cause frayed alignment ends rather than indicating mismatches. An apt trimming algorithm removes these artefacts. Subsequently, the high fidelity consensus of the piled up alignments is generated from a derived frequency matrix. In addition, we compute phred mimicking quality scores and encode positional confidence information in familiar FASTQ format.

long & accurate

Our work flow efficiently integrates 3rd generation read length and 2nd generation accuracy. On genomic and transcriptomic sample data we achieve over 99.95% base call accuracy at a recovery rate of more than 80%. Thus, proovread gives you the best from both worlds.

thomas.hackl@uni-wuerzburg.de

erc
European Research Council
Established by the European Commission

13.1 bwa-proovread

proovread does local score comparison, rather than using a single hard cut-off. bwa-proovread is modified in the same fashion. `proovread.[ch]` extend bwa with an implementation of proovread's binning algorithm. Reporting of alignments is determined by score-comparison within bins. That way repeat alignments are filtered early on, increasing performance and largely reducing disk space requirements.

14 Citing proovread

If you use proovread, please cite:

proovread: large-scale high accuracy PacBio correction through iterative short read consensus. Hackl, T.; Hedrich, R.; Schultz, J.; Foerster, F. (2014).

Please, also recognize the authors of software packages, employed by proovread:

Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. Li H. (2012) (bwa)

Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. Mark J Chaisson; Glenn Tesler. (2012)

SHRiMP: Accurate Mapping of Short Color-space Reads. Stephen M Rumble; Phil Lacroute; Adrian V. Dalca; Marc Fiume; Arend Sidow; Michael Brudno. (2009)

15 Contact

If you have any questions, encounter problems or potential bugs, don't hesitate to contact us. Either report issues on github or write an email to:

- Thomas Hackl - thomas.hackl@uni.wuerzburg.de
- Frank Foerster - frank.foerster@uni-wuerzburg.de