Readjoiner 1.2: User manual.

A fast and memory efficient
string graph-based sequence assembler.

Giorgio Gonnella
Stefan Kurtz

Research Group for Genome Informatics
Center for Bioinformatics
University of Hamburg
Bundesstrasse 43
20146 Hamburg (Germany)

gonnella@zbh.uni-hamburg.de
kurtz@zbh.uni-hamburg.de

18/02/2014

1 Introduction

Readjoiner is a software pipeline for the *de novo* assembly of sequencing readsets, based on the assembly string graph framework [1].

Readjoiner is written in `C` and it is based on the *GenomeTools* library [2]. It has no external library dependencies and may be compiled on any POSIX-compliant operative system. *Readjoiner* is implemented as a collection of tools, all compiled in the single *GenomeTools* binary named `gt`.

The *Readjoiner* assembly pipeline consists in the following phases:

Phase	Tool	Description
Prefiltering	`gt readjoiner prefilter`	encode reads, remove contained and ambiguous
Overlap	`gt readjoiner overlap`	determine all pairs suffix-prefix matches (SPMs)
Assembly	`gt readjoiner assembly`	build the string graph, output the contigs

Readjoiner can be installed either from a binary distribution or from the source code. In both cases, you will have the binary file `gt`. The rest of the manual assumes this binary is in the command line path.

The *Readjoiner* tools are run using the following syntax: `gt readjoiner <tool> <arguments>`, where `<tool>` is either prefilter, overlap or assembly.

The arguments are different, depending on the tool, however some options are common:

- `-help` will show an useful help message on the command line, with a summary of the most important options.
- `-help+` will show an expanded help message, in which further less common options are also listed.
- `-v` increases the verbosity.
- `-q` suppresses all output messages.

1.1 Installation from the source code

For installation from the source code, please read the instruction in the `INSTALL` file (in the main directory of the sources). In most cases a 64-bit version is necessary, so make sure you use the `64bit=yes` make flag. External dependencies such as cairo are not necessary for *Readjoiner*, thus if you plan to use only *Readjoiner* you may safely choose to use the `cairo=no` make flag.

If you use the `amalgamation=yes` make flag, you will probably notice a significative speed-up, thus this option is reccomended. Further improvements in the runtime can be achieved by using `assert=no`: this skips all runtime assertions; if you encounter any problem by running *Readjoiner* then you should (re-)compile using assertions before reporting a bug, as assertions are an useful source of information for the developer.

Using the option `threads=yes` you can enable multithreading support. Currently the overlap phase can be run in parallel mode, thus this option is reccomended if *Readjoiner* is run on a multi-core or multi-processor machine.

2 Error correction

Most sequencing technologies generate reads which contain technology-specific errors. For example, 454 and IonTorrent reads will contain mainly indels in homopolymeric regions, while Illumina reads will contain mainly substitution errors. An error correction prior to genome assembly is an essential step, and it is very important for Readjoiner, that the input reads have been already processed using an error correction tool.

A number of stand-alone error correction tools exists: an example of a tool with a good performance on subsitution errors in Quake (Kelley et al., 2010). For homopolymer errors if a reference sequence is available, have a look to our error corrector HOP, also a part of GenomeTools.

For susbtitution errors there is also an experimental support in Readjoiner of k-mer based error correction. If you want to try it, run `gt dev seqcorrect -help` for more information.

3 Prefiltering

Readjoiner prefiltering step has the following functions:

- Remove ambiguity-containing reads
- Remove redundant information (duplicate and suffix/prefix reads)
- Encode the readset in GtEncseq format

3.1 Describing the read set

The input to *Readjoiner Prefilter* consists of sequencing reads, in FastQ or Fasta format. The input is specified using the `-db` option, which takes multiple arguments, each argument describes a sequencing library.

For single-end libraries, the corresponding argument of `-db` is simply the filename. Paired-end libraries are accepted in two variants: *two-files* libraries, where a file contains the forward reads and a second file the reverse reads in the same order, and *interleaved* libraries, consisting of a single file, where the first, third, fifth, etc sequences are the forward reads, the second, fourth, sixth, etc sequences are the reverse reads. For two-files libraries, the argument of `-db` is in the format `<file1>:<file2>:<mean>[,<std>]` where `<file1|2>` are filenames, `<mean>` is the average insert size and `<std>` is the standard deviation of insert sizes (which can be omitted if unknown). For interleaved libraries the library is specified as `<file>:<mean>[,<std>]`.

Some examples:

Argument of `-db`	Meaning:
se.fas	single-end library (`se.fas`)
pe.fastq:200	interleaved paired-end library with average insert size of 200, standard deviation unknown
pe_f.fastq:pe_r.fastq:10000	a paired-end library in two files, with average insert size of about 10kb, standard deviation unknown
se.fastq pe.fastq:180,20	a single-end library plus an interleaved paired-end library with average insert size of 180, standard deviation 20
short.fastq:100 long_f.fastq:long_r.fastq:10000,1200 se.fastq	an interleaved paired-end library with insert size 100 plus a two-files paired-end library with insert size 10000, standard deviation 1200, and a single-end library

3.2 Setting the read set name

The argument of the option `-readset` is the read set name. This will be used as basename for the output and intermediate files and must specified in subsequent calls to the `overlap` and `assembly` tools.

3.3 Contained reads

Duplicate reads and suffix/prefix reads (i.e. reads which are suffixes or prefixes of other reads) are contained reads, thus redundant sequence information, which is not represented in the string graph [1]. Furthermore, our overlap algorithm assumes that the read set is suffix-prefix-free, thus these reads are recognized and eliminated in the prefiltering phase.

Reads are hereby considered from both strands, thus for example a reads which is equal to the reverse complement of another read is also removed. Internally contained reads (reads equal to an internal substring of other reads / reverse complement) are not identified here, but during the *Readjoiner Overlap* step.

In a paired-end library, if a read is contained, its mate pair is discarded too, to keep the library balanced.

3.4 Quality scores

Readjoiner supports the input formats FastQ and Fasta. The FastQ format is assumed to use the standard *phred33* scores. As an alternative, *phred64* scores are also supported; in this case the option `-phred64` must be specified. The only limitation is that it is not possible to mix libraries with *phred33* scores and libraries with *phred64* scores.

The quality scores can be used for quality-based trimming. In order to activate this feature, the `-maxlow` option must be specified, with an argument n, which is the maximal number of low-quality

bases acceptable in a read. If a read has more than n low-quality bases, it is discarded. The definition of low-quality bases is any base whose associated quality score is at most l, where l is an user-defined parameter, by default 3, which can be set using the `-lowqual` option.

3.5 Ambiguous base calls

Any read containing ambiguities is discarded as low-quality. If you want to keep reads containing ambuigities, you will have to split them at the ambiguity positions or permutate the ambiguities to a random non-ambiguous code using an external program or script. In a paired-end library, if a read is discarded, its mate pair is discarded too, so that the library is still balanced.

4 The overlap phase

To construct the string graph, the suffix-prefix matches (SPM) among all pairs of reads must be calculated. Due to the small size of the DNA alphabet (4), small random matches from reads originating from different regions of the original DNA molecule are common. Thus, to avoid spurious matches a minimal match length pameter ℓ is used.

Transitive edges are not present in the final string graph [1]. *Readjoiner Overlap* matching algorithm allows one to recognize which SPM would correspond to transitive edges in the graph. Thus the graph can be constructed only including the irreducible edges, requiring less memory.

The *Readjoiner Overlap* tool computes the list of suffix-prefix matches among all pairs of reads using a suffix sorting and scanning approach. Non-relevant suffixes are excluded, such as those shorter than ℓ or without an initial k-mer matching a read or reverse complement of read.

An index of the SPM-relevant suffixes is constructed during this phase. The index construction is partitioned in order to limit the space peak. *Readjoiner* uses heuristics to automatically calculates a value for this parameter, which is usually a good choice.. However, a memory limit or number of parts can be given as a parameter to control this time vs. space tradeoff.

If the binary has been compiled with multithreading support, you can specify the number of threads as follows: `gt -j <N> readjoiner overlap <arguments>`.

4.1 Running the *Readjoiner Overlap* tool

The input to the *Readjoiner* overlap tool is a prefix- and suffix-free readset in *GenomeTools* encoded sequence format, usually the output of *Readjoiner Prefilter*.

The following options shall be used:

`-readset` *readset*
: Specify the readset name. The argument should be the same used for the prefilter tool.

`-l` *minlen*
: Specify the minimal SPM length parameter.

The minimal SPM length parameter is an essential parameter of the procedure. A too large value will make the pipeline faster, but potentially miss some important matches. A too small value will make the pipeline too slow, and add too many spurious overlaps.

We found that the ideal value is in the range $\approx 55 - 65$ for 100 bp reads. The assembly phase can be repeated using different values of the minimal SPM length, without repeating the overlap phase, as long as these values are larger than the value used in the overlap phase. Therefore, one can choose a little smaller value, e.g. ≈ 50 during the overlap phase and then test the assembly phase using several values equal or larger than that.

The output of the overlap phase consists in one or more `.spm` files, containing a binary encoded list of the non-redundant irreducible suffix-prefix matches in the readset. It is possible to manually examine the results, by converting this list a text format, using the following command: `gt readjoiner spmtest -readset <readsetname> -test showlist`.

5 The assembly phase

The *Readjoiner Assembly* tool constructs a string graph. Each non-contained read in the sequencing readset is represented by a pair of vertices modeling its two extremities (Begin and End vertices). Irreducible SPMs are represented by directed edges.

Sequencing errors present in the reads lead to characteristic paths in the graph. The error-correction algorithms described in [4] are implemented and can be applied to recognize dead ends and p-bubbles and remove them (option `-errors`). Finally the graph is traversed and sequences corresponding to unbranched paths are output.

5.1 Running the the *Readjoiner Assembly* tool

The input consists in SPM lists in *Readjoiner* format which are the output of *Readjoiner Overlap*.

The following options shall be used:

`-readset` *readset*
: Specify the readset name, This must be the same used in the overlap phase.

`-l` *minlen*
: (optional) Specify the minimal SPM length parameter. The value should be at least as high as the one used for *Readjoiner Overlap*. If an higher value is used, shorter overlaps are not loaded in the string graph.

`-errors`
: Clean short dead ends and remove p-bubbles. This is reccomended

`-spmfiles` *n*
: If the overlap phase was run using muliple threads, this option is mandatory and should be set to the same number of threads used by the

`-j` i
: n the overlap phase.

6 Examples

Consider a dataset containing paired-end reads with an insert size of 1000bp (st.dev. 120bp) in the files `p_f.fastq` and `p_r.fastq` and single-end reads in the file `se.fastq`.

This example assumes the reads have been previously error-corrected, otherwise the results will be probably very poor.

The first step will be run by *Readjoiner Prefilter*.

```
> gt readjoiner prefilter -readset myreadset \
  -db p_f.fastq:p_r.fastq:1000,120 se.fastq
```

The result is output in *GenomeTools* encoded sequence format and consist in a `readset.esq` file. The encoded sequence can now be used as input for *Readjoiner Overlap*. We will use a minimal SPM lenght of 50 for this example and suppose *Readjoiner* has been compiled with multi-threading support and 8 cores are available.

```
> gt -j 8 readjoiner overlap -readset readset -l 50
```

The output will be a set of files `readset.<N>.spm`, with N in the range 0 to 7, containing lists of irreducible non-redundant SPMs in *Readjoiner* binary format. Note that the `-j` option is specified in a different position (between `gt` and `readjoiner`) than the other options.

The next step consists in running the assembly phase using the *Readjoiner Assembly* tool.

```
> gt readjoiner assembly -spmfiles 8 -readset readset -l 60
```

The `-spmfiles` option is only necessary if the overlap phase was run using multiple threads: we used the same value for `-spmfiles` which was used for the `-j` option in the overlap phase.

We used an higher mimimal length value that the one used in the overlap phase. If the results are not satisfying, you can repeat the assembly phase using a smaller value, such as `-l` *50* or `-l` *55* (the minimum that can be used is the value used in the overlap phase) and a larger value, such as `-l` *65* or `-l` *70*.

The results are contigs, which are saved in FASTA format in `readset.contigs.fas`. Basic statistics of the assembly results, such as the N50 value and the total length, are displayed in the standard output.

References

[1] Myers, EW. (2005). The fragment assembly string graph. *Bioinformatics*, 21 Suppl 2, 79–85.

[2] Gremme, G. (2011). The GENOMETOOLS genome analysis system. `http://genometools.org`.

[3] Abouelhoda MI, Kurtz S, and Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2:53–86, 2004.

[4] Hernandez D, François P, Farinelli L, Osterås M, and Schrenzel J. (2008). De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res*, 18(5), 802–809.

[5] Gonnella G and Kurtz S (2012). Readjoiner. A fast and memory efficient string graph-based sequence assembler. *BMC Bioinformatics*, 13:82.