genomediff user manual

Dirk Willrodt

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

willrodt@zbh.uni-hamburg.de

September 12, 2015

This Manual

Some text is highlighted by different fonts according to the following rules.

- `Typewriter font` is used for the names of software tools.
- `Small typewriter font` is used for file names.
- `Footnote sized typewriter font` with a leading `'-'` is used for program options.
- *small italic font* is used for the argument(s) of an option.

1 Introduction

This document describes *genomediff*, a software tool for measuring evolutionary distances between sets of closely related genomes. These distances are Jukes-Cantor corrected divergence between the pairs of genomes, that is, the number of mutations per base between them.

This distance is called K_r and is based on so called *shustrings* [1, 2, 3]. The calculation of all pairwise distances is alignment free, but the resulting distances have the same biological meaning as if calculated with a multiple sequence alignment.

This software is only able to process closely related distances, because K_r is only reliable for distances < 0.5.

genomediff is written in C and it is based on the *GenomeTools* library [4]. It is called as part of the single binary named `gt`.

The source code can be compiled on 32-bit and 64-bit platforms without making any changes to the sources.

2 Building *genomediff*

As *genomediff* is part of the *GenomeTools* software suite, a source distribution of *GenomeTools* must be obtained, e.g. via the *GenomeTools* home page (`http://genometools.org`), and decompressed into a source directory:

```
$ tar -xzvf genometools-X.X.X.tar.gz
$ cd genometools-X.X.X
```

Where `X.X.X` denotes the desired gt version.

Then, it suffices to call `make` to compile the source using the provided makefile.

It is recommended to use the 64bit-version of the *GenomeTools* executable if your system supports this. Pass the option `64bit=yes` to enable 64 bit support.

The option `amalgamation=yes` allows the compiler to use better optimization.

```
$ make 64bit=yes amalgamation=yes
```

After successful compilation, the *GenomeTools* executable containing *genomediff* is available in the `bin` subfolder of the root directory of the uncompressed source. It can then be installed for system-wide use as follows (do this as root):

```
$ make 64bit=yes amalgamation=yes install
```

Make sure to use the same options as for the compilation step when using the install target!

If a `prefix=<path>` option is appended to this line, a custom directory can be specified as the installation target directory, e.g.

```
$ make 64bit=yes amalgamation=yes install prefix=/home/user/gt
```

will install the `gt` binary in the `/home/user/gt/bin` directory. Please also consult the `README` and `INSTALL` files in the root directory of the uncompressed source tree for more information and troubleshooting advice.

3 Usage

3.1 *genomediff* command line options

Since *genomediff* is part of *GenomeTools*, it is invoked as follows:

`gt genomediff [options] (INDEX | -indexname` *NAME* `SEQFILE SEQFILE [...])`

where `INDEX` is the path without file extension of an encoded sequence containing the genomes to be compared and *NAME* is a name for an encoded sequence to be built from the given `SEQFILES`.

A short description of all possible options is given in Table 1.

Listing 1: Example unitfile: The section '`units`' is mandatory, '`genome1/2`' are examples of names, filenames are paths as given on the command line or during index construction.

```
units = {
  genome1 = { "file1.fas", "file2.fas" },
  genome2 = { "path/file3.fas", "file4.fas" }
}
```

3.2 Input files

The tool *genomediff* can handle three types of prepared indices. The first is an encoded sequence, which can be prepared by *encseq*. Given an encoded sequence, *genomediff* will build an enhanced suffix array in memory and calculate K_r using that index. Second is an enhanced suffix array prepared by the tool *suffixerator* (see `gt suffixerator -help`) and third a compressed FM-index build by the tool *packedindex* (see `gt packedindex mkindex -help`. The usage of FM-indices is not recommended, because calculation of K_r takes significantly longer.

Table 1: *genomediff* command line options

Input options	
`-indextype` *type*	Specify type of index, one of: *esa*|*pck*|*encseq*. Where encseq is an encoded sequence and an enhanced suffix array will be constructed only in memory. default: *encseq*
`-unitfile` *filename*	Specifies genomic units, see below for description. default: undefined
Output options	
`-indexname` *name*	Basename of encseq to construct. default: undefined
ESA options	
`-mirrored`	Virtually append the reverse complement of each sequence default: *no*
`-pl` *n*	Specify prefix length for bucket sort recommendation: use without argument; then a reasonable prefix length is automatically determined. default: *0*
`-dc` *n*	Specify difference cover value. default: *0*
`-memlimit` *n*	Specify maximal amount of memory to be used during index construction (in bytes, the keywords 'MB' and 'GB' are allowed). default: undefined
Miscellaneous options	
`-v`	Be verbose. default: *no*
`-help`	Display help for basic options and exit.
`-help+`	Display help for all options and exit.
`-version`	Display version information and exit.

Another way is to give the names of sequence files directly. Option `--indexname` is mandatory in this case. The given name will be used to store an encoded sequence on disk. File format can be any sequence format supported by *GenomeTools*.

Either way, each given sequence file will be regarded as one genomic unit, regardless of the number of sequences inside that file.

To give the genomic units other names than the filenames or to combine files to single genomic units one can give a unitfile with option `--unitfile`. The format of an example unitfile is shown in Listing 1.

3.3 Output

The output on the standard output stream consists of a line with the number of genomes or units that were compared. It is followed by a quadratic matrix of pairwise distances where each line consists of a file- or unitname and tabulator separated distance values.

Depending on the options of the `gt` call there can be additional output where each line is prefixed by '`#` ' and additional output prefixed by'`debug:` ' on the standard error stream.

4 Example

This section describes two example scenarios, the first being the comparison of multiple genomes organised in separate multiple FASTA-files and the second being the comparison of two genomes consisting of multiple files each.

4.1 Compare genomes in separate files

Consider three files `genome1.fas`, `genome2.fas` and `genome3.fas` each of which could contain multiple FASTA entries. Our machine has 2 GiB RAM. Assuming the index construction would need 5 GiB, we need to split it in at least three parts of equal size or restrict maximal memory requirements.

The simplest way to calculate the distance matrix for these three genomes would be to call:

```
gt genomediff -indexname 3genomes \
              -memlimit 1500MB    \
              genome1.fas genome2.fas genome3.fas
```

`--memlimit` should be reasonable less than available main memory.

This will output the distance matrix on the terminal and store an encoded sequence with basename `3genomes` in the current directory.

In order to save the results to a file use terminal redirection: `gt genomediff ... > outfile`.

The file `outfile` might look like this:

```
3
genome1.fas 0.000000 0.115125 0.267473
genome2.fas 0.115125 0.000000 0.293082
genome3.fas 0.267473 0.293082 0.000000
```

This tabulator separated table can be used for example with *Phylip* or *R* to calculate a phylogenetic tree.

Another way to calculate the same distances if an enhanced suffix array of the given files with name `3genomes_idx` already exists on disk would be like this:

```
gt genomediff -indextype esa 3genomes_idx > outfile
```

To reuse an existing encoded sequence just give the basename of it:

```
gt genomediff 3genomes > outfile
```

4.2 Compare two genomes in multiple files

Assume we have two genomes that consist of multiple chromosomes in separate files. For example, genome1 consists of `g1_chr1.fas` and `g1_chr2.fas` while the two files for genome2 are named accordingly. The unitfile could be organized like this:

```
units = {
  genome1 = { "g1_chr1.fas", "g1_chr2.fas" },
  genome2 = { "g2_chr1.fas", "g2_chr2.fas" }
}
```

The name of the unitfile in our example will be `units`.

Now we could call *genomediff* like this:

```
gt genomediff -indexname 2genomes \
              -unitfile units     \
              g1_chr1.fas g1_chr2.fas g2_chr1.fas g2.chr2.fas > output
```

File `output` could look like this:

```
2
genome1 0.000000 0.115125
genome2 0.115125 0.000000
```

Bibliography

References

[1] Bernhard Haubold, Mirjana Domazet-Loso, and Thomas Wiehe. An alignment-free distance measure for closely related genomes. In *RECOMB-CG '08: Proceedings of the international workshop on Comparative Genomics*, pages 87–99, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] B. Haubold, P. Pfaffelhuber, M. Domazet-Loso, and T. Wiehe. Estimating mutation distances from unaligned genomes. *J. Comput. Biol.*, 16:1487–1500, "Oct" 2009.

[3] Bernhard Haubold, Floyd A. Reed, and Peter Pfaffelhuber. Alignment-free estimation of nucleotide diversity. *Bioinformatics*, 27(4):449–455, 2011.

[4] Gordon Gremme. The GenomeTools genome analysis system. `http://genometools.org`.