The Journal of Open Source Software

GFAKluge: A C++ library and command line utilities for the Graphical Fragment Assembly formats

Eric T. Dawson[1, 2, 3] **and Richard Durbin**[2, 3]

1 Division of Cancer Epidemiology and Genetics, National Cancer Institute, Rockville, MD, USA **2** Department of Genetics, University of Cambridge, Cambridge, UK **3** Wellcome Sanger Institute, Hinxton, UK

DOI: 10.21105/joss.01083

Software

- Review
- Repository
- Archive

Submitted: 24 September 2018
Published: 22 January 2019

Summary

GFAKluge is a set of command line utilities and a C++ library for parsing and manipulating the Graphical Fragment Assembly (GFA) format. Genome assembly algorithms often use graph structures to represent relationships between reads during the assembly process. Such information is typically thrown away when assemblies are converted to FASTA files of contig sequences. Previous attempts to convey graph information did not gain widespread acceptance because there were no standard representations that were easily parsed and extensively used. The Graphical Fragment Assembly (GFA) format was proposed as a way to encode the graph structure of an assembly in a human-readable text format (Li, 2014). GFA aims to provide a single format for interchange between software for assembly, scaffolding, assessment and visualization. Such programs are often written in high-performance programming languages such as C or C++. GFAKluge facilitates interprogram exchange by providing a high-level C++ API for developers and a set of command line tools for users. We hope the availability of an open-source, easily extensible API will encourage software developers to consider adding support for GFA to their bioinformatics programs.
Homepage: https://github.com/edawson/gfakluge *License:* MIT

Command Line Utilities

GFAKluge also provides a command line interface for working with GFA. This includes support for common tasks on assemblies such as calculating assembly N50 or graph statistics. There are also methods for merging assemblies, reformating files for readability, and converting between the GFA 1.0 and GFA 2.0 specifications. A tool for constructing basic variation graphs from a FASTA file and a VCF file is also included. Many other tools exist for manipulating the GFA formats (Myers, Jackman, Gonnella, Chin, & Durbin, 2015), though only RGFA (Gonnella & Kurtz, 2016), GfaPy (Gonnella & Kurtz, 2017) and ABySS2.0 (S. D. Jackman et al., 2017) are known to produce and consume both versions. By allowing interconversion between the compatible subsets of the formats, the `gfak convert` tool allows programs that usually can't communicate to share data without changes to their code. We have used GFAKluge to convert GFA from TwoPaCo (Minkin, Pham, & Medvedev, 2016) for visualization in Bandage (Wick, Schultz, Zobel, & Holt, 2015), to calculate assembly statistics from the Falcon assembler (C.-S. Chin et al., 2016), and to extract FASTA from a `vg msga` assembly (Garrison et al., 2018).

```
# Convert GFA 2.0 from TwoPaCo to GFA 1.0 for ingestion by Bandage.
gfak convert -S 1.0 data/gfa_2.gfa

# Calculate assembly statistics
gfak stats -a data/gfa_2.gfa

# Extract FASTA entries from a GFA file
gfak extract data/gfa_2.gfa
```

The full list of `gfak` commands follows:

```
convert: Convert between GFA 0.1 <-> 1.0 <-> 2.0
diff:    Determine whether two GFA files have identical graphs
extract: Convert the S lines of a GFA file to FASTA format.
fillseq: Add sequences from a FASTA file to S lines.
ids:     Coordinate the ID spaces of multiple GFA graphs.
concat:  Merge GFA graphs (without ID collisions).
sort:    Print a GFA file in HSLP / HSEFGUO order.
stats:   Get assembly statistics (e.g. N50) for a GFA file.
subset:  Extract the subgraph between two IDs in a graph.
trim:    Remove elements from a GFA graph.
```

Examples of most commands are included in the examples.md file.

Integrating GFAKluge into an existing program

As an example of how to use the GFAKluge API, we briefly summarize its use in the variation graph toolkit vg (Garrison et al., 2018). vg creates bidirected sequence graphs from assemblies and population variation that can then be used for read mapping and variant calling. We incorporated GFAKluge into vg to support input and output of GFA. Reading in a GFA file requires one line of code and is agnostic to the GFA version used. Converting from GFA to vg's internal structures and vice versa requires approximately forty lines of code. Changing output from GFA v1.0 to GFA v2.0 requires a single API call. This allows vg to take assemblies in GFA format from TwoPaCo and many other assembly algorithms. The gfak command line tools can be used to calculate assembly graph statistics on graphs produced by vg. A full description of the developer API is available in the interface.md file.

Chin, C.-S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., et al. (2016). Phased diploid genome assembly with single-molecule real-time sequencing. *Nature Methods*, *13*, 100–1054. doi:doi:10.1038/nmeth.4035

Garrison, E., Sirén, J., Novak, A. M., Hickey, G., Eizinga, J. M., Dawson, E. T., Jones, W., et al. (2018). Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, *36*, 875–879. doi:doi:10.1038/nbt.4227

Gonnella, G., & Kurtz, S. (2016). RGFA: Powerful and convenient handling of assembly graphs. *PeerJ*, *4*. doi:doi:10.7717/peerj.2681

Gonnella, G., & Kurtz, S. (2017). GfaPy: A flexible and extensible software library for handling sequence graphs in python. *Bioinformatics*, *33*, 3094–3095. doi:doi:10.1093/bioinformatics/btx398

Jackman, S. D., Varndervalk, B. P., Mohamadi, H., Chu, J., Yeo, S., Hammond, S. A., Jahesh, G., et al. (2017). ABySS 2.0: Resource-efficient assembly of large genomes using a bloom filter. *Genome Research*, *27*, 768–777. doi:doi:10.1101/gr.214346.116

Li, H. (2014). A proposal of the grapical fragment assembly format. Retrieved from http://lh3.github.io/2014/07/19/a-proposal-of-the-grapical-fragment-assembly-format

Minkin, I., Pham, S., & Medvedev, P. (2016). TwoPaCo: An efficient algorithm to build the compacted de bruijn graph from many complete genomes. *Bioinformatics*, *33*, 4024–4032. doi:doi:10.1093/bioinformatics/btw609

Myers, G., Jackman, S., Gonnella, G., Chin, J., & Durbin, R. (2015). Graphical fragment assembly (gfa) format specification. Retrieved from https://github.com/GFA-spec/GFA-spec

Wick, R., Schultz, M., Zobel, J., & Holt, K. (2015). Bandage: Interactive visualisation of de novo genome assemblies. *Bioinformatics*, *31*, 3350–3352. doi:doi:10.1093/bioinformatics/btv383