Bioinformatics, 38(17), 2022, 4214–4216
https://doi.org/10.1093/bioinformatics/btac460
Advance Access Publication Date: 7 July 2022
Applications Note

Genome analysis

Gfastats: conversion, evaluation and manipulation of genome sequences using assembly graphs

Giulio Formenti [1,*], Linelle Abueg[1], Angelo Brajuka[1], Nadolina Brajuka[1], Cristóbal Gallardo-Alba[2], Alice Giani[3], Olivier Fedrigo[1] and Erich D. Jarvis[1,4]

[1]The Vertebrate Genome Laboratory, The Rockefeller University, New York, NY 10065, USA, [2]Bioinformatics Group, Department of Computer Science, Albert-Ludwigs-University Freiburg, Freiburg 79110, Germany, [3]Helen and Robert Appel Alzheimer Disease Research Institute, Feil Family Brain and Mind Research Institute, Weill Cornell Medicine, New York, NY 10021, USA and [4]Howard Hughes Medical Institute, Chevy Chase, Maryland 20815, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on March 8, 2022; revised on May 28, 2022; editorial decision on July 2, 2022; accepted on July 6, 2022

Abstract

Motivation: With the current pace at which reference genomes are being produced, the availability of tools that can reliably and efficiently generate genome assembly summary statistics has become critical. Additionally, with the emergence of new algorithms and data types, tools that can improve the quality of existing assemblies through automated and manual curation are required.

Results: We sought to address both these needs by developing gfastats, as part of the Vertebrate Genomes Project (VGP) effort to generate high-quality reference genomes at scale. Gfastats is a standalone tool to compute assembly summary statistics and manipulate assembly sequences in FASTA, FASTQ or GFA [.gz] format. Gfastats stores assembly sequences internally in a GFA-like format. This feature allows gfastats to seamlessly convert FAST* to and from GFA [.gz] files. Gfastats can also build an assembly graph that can in turn be used to manipulate the underlying sequences following instructions provided by the user, while simultaneously generating key metrics for the new sequences.

Availability and implementation: Gfastats is implemented in C++. Precompiled releases (Linux, MacOS, Windows) and commented source code for gfastats are available under MIT licence at https://github.com/vgl-hub/gfastats. Examples of how to run gfastats are provided in the GitHub. Gfastats is also available in Bioconda, in Galaxy (https://assembly.usegalaxy.eu) and as a MultiQC module (https://github.com/ewels/MultiQC). An automated test workflow is available to ensure consistency of software updates.

Contact: giulio.formenti@gmail.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

In recent years, we have witnessed an unprecedented increase in the number of publicly available genomes (Lewin *et al.*, 2018). Thanks to advancements in genome sequencing and assembly (Rhie *et al.*, 2021) many of these genomes are more accurate and contiguous. Reference genomes are made available in publicly maintained archives such as GenBank by the US National Center for Biotechnology Information (NCBI, www.ncbi.nlm.nih.gov/genbank), the European Nucleotide Archive (ENA, www.ebi.ac.uk/ena/browser/home) by the European Bioinformatics Institute, the DNA Data Bank of Japan (DDBJ, www.ddbj.nig.ac.jp), the China National GeneBank (CNGB, https://db.cngb.org), or project-related repositories such as the Vertebrate Genomes Project (VGP) Genome Ark (https://vgp.github.io/) (Rhie *et al.*, 2021). Assemblies are usually stored as collections of sequences representing either contigs (i.e. contiguous stretches of nucleotide sequences) or scaffolds (i.e. contigs separated by gaps of unknown sequence). The size of gaps can be approximately estimated (sized gaps) or unknown.

Sequence collections are generally stored in the popular FASTA format, developed in 1985 (Lipman and Pearson, 1985). In FASTA, each sequence is introduced by a '>' character followed by a header and a comment, and the sequence on newlines. Like FASTA, the FASTQ format was developed over two decades ago at the Wellcome Trust Sanger Institute (Cock *et al.*, 2010) and later popularized by Illumina to store short-read sequencing data with per-base quality information. More recently, the representation of biological sequences has been expressed under the conceptual framework of graph theory (Paten *et al.*, 2017). In a graph, genome assemblies can be represented as collections of sequences (nodes) linked by experimental evidence

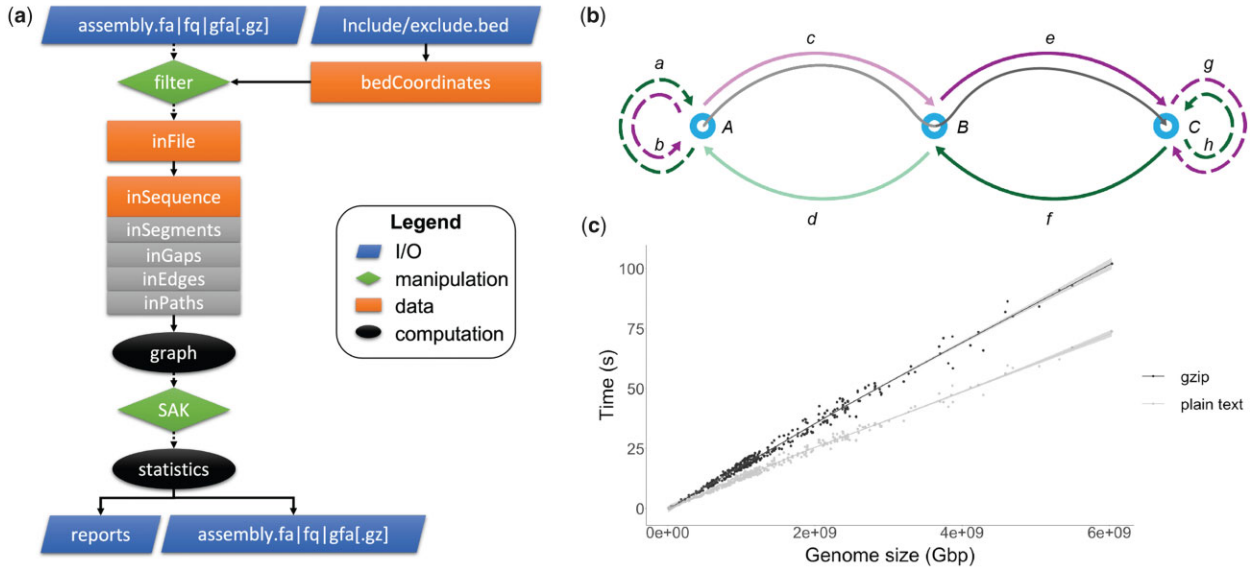

Fig. 1. (**a**) Schematic of gfastats workflow. Inputs (top trapezoids) include genome assemblies in FASTA, FASTQ, GFA [.gz] formats and include/exclude lists as bed coordinate files for filtering (first diamond). These are represented internally by multiple C++ classes including their constituent elements (rectangles). The assembly can be converted to a graph (first oval), to ease manipulation by the internal Swiss Army Knife (SAK; second diamond), and then summary statistics are computed (second oval). A variety of outputs can be generated, such as summary statistics and new sequences in *fa* format. (**b**) Internal bidirected graph representation of the input sequences. Segments (nodes A, B, C) are connected by forward (b, c, e, g) or backward (a, d, f, h) gaps edges. Terminal nodes can optionally be associated with gaps (dashed lines, gap edges a, b, g and h). An assembly scaffold is a path in the graph (e.g. A → c → B → e → C, grey middle line). Sequence manipulation can be achieved using the internal SAK. For instance, the given path could be split by removing gap edges c and d that connect segment nodes A and B, leading to a disconnected node A, and two connected nodes B and C linked by edges e and f (portion of the path in light grey removed). Overlap edges can be treated in the same way. (**c**) Evaluation of gfastats runtime. Performance time is a function of genome size, with gfastats runtime increasing linearly. There is a small increase in time when handling gzip-compressed files

(edges). GFA is a popular format to store sequence data as graphs. GFA1 (http://gfa-spec.github.io/GFA-spec/GFA1.html), which was introduced in 2014 (http://lh3.github.io/2014/07/19/a-proposal-of-the-grapical-fragment-assembly-format), can be used to conveniently store and visualize (Wick *et al.*, 2015) key features of sequence graphs, such as the product of an assembly (Cheng *et al.*, 2021), the representation of variation in genomes or overlaps between reads. Since the graph is not yet collapsed to a linear representation, many additional characteristics can be deduced. Of particular importance is the residual graph connectivity, such as the presence of ambiguous overlaps originating from repetitive elements or the presence of bubbles originating from heterozygosity, both leading to reduced contiguity if not resolved.

The GFA1 format consists of lines with tab-delimited fields. The first field defines the line type, which in turn defines additional required fields, followed by optional fields. Examples of line types are segments (S, usually a contig) and edges (L, usually an overlap between two contigs). GFA was later generalized to GFA2, which allows specifying an assembly graph in either less detail (e.g. only the topology of the graph) or in more detail (e.g. the multi-alignment of reads underlying each sequence) (https://github.com/GFA-spec/GFA-spec/blob/master/GFA2.md). Importantly, GFA2 introduces more line types to include gaps (G), allowing scaffolds (i.e. contig separated by gaps) to be represented.

As more and more reference genomes become available, a single fast, versatile tool that can compute assembly summary statistics from a variety of file formats is warranted. In the framework of the VGP, which aims to generate high-quality genome assemblies for all vertebrate species, we have developed and present here gfastats (short for graph-based *FA* statistics). By internally representing any input sequence (FASTA, FASTQ, GFA1/2) in a more general GFA2-like format, gfastats can efficiently compute accurate summary statistics. It further allows simultaneous manipulation of the assembly sequences, thereby potentially facilitating both automated assembly and manual curation.

2 Results

For optimization purposes, gfastats is coded solely in C++, taking full advantage of object-oriented programming. In gfastats v1.2.0 (the version presented hereinafter), contigs (segments), edges and gaps are represented with classes, and so are the collections of paths through contigs and gaps that, taken together, represent a genome assembly (Fig. 1a). Features of interest are represented using bed coordinates. Input includes any *fa* (FASTA, FASTQ, GFA [.gz]) file. Since gfastats reads and stores any input in a GFA-like format, it allows the seamless conversion between different formats (FASTA<>FASTQ<>GFA[.gz]). Inputs are processed on the fly to generate summary statistics.

Gfastats computes a growing number of assembly/sequence metrics (Fig. 1a;). We compared gfastats to QUAST Gurevich et al. (2013) and SeqKit Shen *et al.* (2016) and found that gfastats provides the most complete set of reference-metrics (Supplementary Table S1). Gfastats summary statistics are available also as a MultiQC module (Ewels *et al.*, 2016). Metrics for each contig can be generated as well. AGP (A Golden Path), BED coordinates and sizes of scaffolds, contigs and gaps can be conveniently outputted. Input can be filtered in a pre-processing step to include/exclude sequences or portions of them using scaffold lists or bed coordinate files. Sequences can be sorted, either according to a list or to other characteristics (name, length, etc.). Gfastats also allows homopolymer compression and decompression, a feature increasingly useful when dealing with long reads.

Since the assembly process is still imperfect, manipulation of contig and scaffold sequences is also needed. High-quality genome assemblies often require a long process of curation, in which experts manually validate and correct the assembly using evidence from the raw data (Howe *et al.*, 2021). The process also relies on file format specifications not adapted and not specifically designed for this purpose. By representing any input sequence as a graph, gfastats allows their manual manipulation. For instance, gfastats can build a bidirected graph representation of the assembly using adjacency lists, where each node is a segment, and each edge is a gap (Fig. 1b). Canonical algorithms (e.g. Depth First Search) are used to walk the graph. In this case, the manipulation is achieved by the internal 'swiss army knife' (SAK) for genome assembly. SAK evaluates a set of basic sequential instructions, i.e. actions to be performed one-by-one on the graph to manipulate the sequence (e.g. join or split contigs, reverse complement sequence, etc.). Here, the representation of the assembly as a graph allows several operations to be performed (e.g. the removal of all trailing Ns from scaffolds by dropping all terminal gap edges). Once all instructions for the SAK are processed, metrics are updated and returned, allowing evaluation of the

revised assembly. The filtered and/or manipulated input can also be outputted in any *fa* format, thereby generating new sequences.

Testing on a 2.8 GHz Quad-Core Intel Core i7 using 370 genome assemblies (both primary and alternate) from the VGP shows that gfastats can compute all summary statistics in less than a minute for genome assemblies of size up to 4 Gbp in O(N) time (Fig. 1c). Assembly manipulation comes with minimal overhead.

3 Discussion and future perspectives

As graph representations of genome assemblies become more popular Jarvis et al. (2022); Cheng et al. (2022); Rautiainen et al. (2022), effective tools that make assembly graph storage, analysis and manipulation easily accessible become necessary. While a few libraries already exist to deal with GFA files (Dawson and Durbin, 2019) (https://github.com/lh3/gfatools), they do not make FASTA and GFA fully interoperable, and do not directly allow their seamless manipulation. The design of gfastats addresses this need in a modular framework, allowing new features to be readily implemented. Potential new features include: file indexing to test multiple hypotheses with minimal runtime overhead, pattern search, sequence soft/hard-masking and new instructions to the SAK. Additional FASTA and GFA statistics can also be introduced based on the needs of the genomics community. Importantly, gfastats introduces a whole new conceptual framework for assembly manipulation where the results of automated algorithms or manual curation be integrated in a single file format and can be expressed in a human-readable set of instructions for the SAK, which also conveniently acts as a log of the changes that were applied during the process.

Acknowledgements

We thank Björn Grüning for helping with the implementation of gfastats in Conda and Galaxy.

Funding

G.F., L.A., N.B., O.F. and E.D.J. were supported by the Rockefeller University and HHMI funds. C.G. was supported by DataPLANT 442077441 through the German National Research Data Initiative (NFDI 7/1).

Conflict of Interest: none declared.

Data availability

All VGP assemblies used for evaluation are publicly available through the Genomeark (https://vgp.github.io)}.

References

Cheng,H. *et al.* (2021) Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nat. Methods*, **18**, 170–175.

Cheng,H. *et al.* (2022) Haplotype-resolved assembly of diploid genomes without parental data. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-022-01261-x.

Cock,P.J.A. *et al.* (2010) The sanger FASTQ file format for sequences with quality scores, and the solexa/illumina FASTQ variants. *Nucleic Acids Res.*, **38**, 1767–1771.

Dawson,E.T. and Durbin,R. (2019) GFAKluge: a C++ library and command line utilities for the graphical fragment assembly formats. *J. Open Source Softw.*, **4**, 1083.

Ewels,P. *et al.* (2016) MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, **32**, 3047–3048.

Gurevich,A. *et al.* (2013) QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, **29**, 1072–1075.

Howe,K. *et al.* (2021) Significantly improving the quality of genome assemblies through curation. *Gigascience*, **10**, giaa153.

Jarvis,E.D. *et al.* (2022) Automated assembly of high-quality diploid human reference genomes. *bioRxiv*. https://doi.org/10.1101/2022.03.06.483034.

Lewin,H.A. *et al.* (2018) Earth BioGenome project: sequencing life for the future of life. *Proc. Natl. Acad. Sci. USA*, **115**, 4325–4333.

Lipman,D.J. and Pearson,W.R. (1985) Rapid and sensitive protein similarity searches. *Science*, **227**, 1435–1441.

Paten,B. *et al.* (2017) Genome graphs and the evolution of genome inference. *Genome Res.*, **27**, 665–676.

Rautiainen,M. *et al.* (2022) Verkko: telomere-to-telomere assembly of diploid chromosomes. *bioRxiv*. https://doi.org/10.1101/2022.06.24.497523.

Rhie,A. *et al.* (2021) Towards complete and error-free genome assemblies of all vertebrate species. *Nature*, **592**, 737–746.

Shen,W. *et al.* (2016) SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation. *PLoS One*, **11**, e0163962.

Wick,R.R. *et al.* (2015) Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*, **31**, 3350–3352.