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

1The Vertebrate Genome Laboratory, The Rockefeller University, New York, NY 10065, USA, 2Bioinformatics Group, Department of Computer Science, Albert-Ludwigs-University Freiburg, Freiburg 79110, Germany, 3Helen and Robert Appel Alzheimer Disease Research Institute, Feil Family Brain and Mind Research Institute, Weill Cornell Medicine, New York, NY 10021, USA and 4Howard 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 FASTA* 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 (Rhee 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://cnb.cbcb.umd.edu), or project-related repositories such as the Vertebrate Genomes Project (VGP) Genome Ark (https://vgp.github.io/) (Rhee 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...
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 (Etter 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 (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., 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 Genomark (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.