Bioinformatics, 2025, **41(7)**, btaf388
https://doi.org/10.1093/bioinformatics/btaf388
Advance Access Publication Date: 7 July 2025
Applications Note

OXFORD

Genome analysis

Agptools: a utility suite for editing genome assemblies

Edward S. Ricemeyer[1,2,*], Rachel A. Carroll[1], Wesley C. Warren[1,3]

[1]Bond Life Sciences Center, University of Missouri, Columbia, MO 65221, United States
[2]Palaeogenomics Group, Institute of Palaeoanatomy, Domestication Research and the History of Veterinary Medicine, Ludwig-Maximilians-Universität München, Munich 80539, Germany
[3]Division of Animal Sciences, Department of Surgery, University of Missouri, Columbia, MO 65201, United States

*Corresponding author. Bond Life Sciences Center, University of Missouri, 1201 Rollins St., Columbia, MO 65201, United States. E-mail: edward.rice@lmu.de.

Associate Editor: Christina Kendziorski

Abstract

Summary: The AGP format is a tab-separated table format describing how components of a genome assembly fit together. A standard submission format for genome assemblies is a fasta file giving the sequence of contigs along with an AGP file showing how these components are assembled into larger pieces like scaffolds or chromosomes. For this reason, many scaffolding software pipelines output assemblies in this format. However, although many programs for assembling and scaffolding genomes read and write this format, there is currently no published software for making edits to AGP files when performing assembly curation. We present agptools, a suite of command-line programs that can perform common operations on AGP files, such as breaking and joining sequences, inverting pieces of assembly components, assembling contigs into larger sequences based on an AGP file, and transforming between coordinate systems of different assembly layouts. Additionally, agptools includes an API that writers of other software packages can use to read, write, and manipulate AGP files within their own programs.

Availability and implementation: Source code and binaries freely available for download at https://github.com/WarrenLab/agptools, implemented in Python and supported on all operating systems.

1 Introduction

New technologies for genome sequencing and new algorithms for genome assembly have made it possible to assemble genomes to a chromosome level at a low cost (Rice and Green 2019, Rhie *et al.* 2021). The availability of highly accurate long reads (Cheng *et al.* 2022), as well as ultra-long reads (Jain *et al.* 2018), has even made gap-free telomere-to-telomere genome assemblies possible for an increasing number of species (Belser *et al.* 2021, Bliznina *et al.* 2021, Xue *et al.* 2021, Nurk *et al.* 2022, Rautiainen *et al.* 2023, Olagunju *et al.* 2024). Now, more and more species have many different genome assemblies available, which are then combined into pangenome graphs to provide advantages such as decreased reference bias, increased mapping accuracy, and better recovery of structural variation existing within the species (Ebler *et al.* 2022, Leonard *et al.* 2023, Smith *et al.* 2023, Groza *et al.* 2024).

However, although assembling high-quality genomes has become easier and more cost-effective over time, most assemblies today are a result of combining several types of data using a sequence of different software packages in a pipeline, and require human curation to correct errors between the steps. For example, a current best-practice genome assembly pipeline involves first assembling long reads into contigs, then combining the contigs into scaffolds using one or more long-range data sources such as optical mapping or proximity ligation, and finally assigning scaffolds into chromosomes based on either a physical map or synteny with another species, with manual curation occurring between steps to correct errors (Rhie *et al.* 2021). The newer technique of telomere-to-telomere assembly uses programs that output an assembly graph rather than a linear assembly, which the user then has to curate (Rautiainen *et al.* 2023, Cheng *et al.* 2024).

Scaffolders such as SALSA (Ghurye *et al.* 2019), YaHS (Zhou *et al.* 2023), and RagTag (Alonge *et al.* 2022) all output an AGP (A Golden Path) file that describes how the input sequences are combined and arranged into the larger output sequences. AGP is a tabular format where each line describes a segment of sequence in the output assembly, using coordinates of the input assembly (AGP Specification v2.1. 2025). For example, if an output scaffold is composed of two input contigs separated by a gap, the AGP file would use three lines to describe this scaffold: first, a line giving the coordinates and orientation of the first input contig in the output scaffold; next, a line describing the gap, including what evidence was used to infer the order and orientation of the sequences flanking the gap; and finally, a line describing the second input contig. These three lines, along with the sequences of the two contigs, thus contain all the information necessary to construct the resulting scaffolds.

The AGP file format has become a standard way of describing the layout of a genome assembly, and is the format for genome assembly submissions to both NCBI and Ensembl assembly databases. Representing an assembly as a set of contiguous sequences along with a description of how they are ordered and oriented into chromosomes has several advantages over representing it as a set of final sequences containing gaps represented as strings of N. An AGP contains more

Received: 26 March 2025; **Revised:** 17 June 2025; **Editorial Decision:** 1 July 2025; **Accepted:** 3 July 2025

information, in the form of gap line evidence fields, about what data were used to join pairs of neighboring sequences, which is especially relevant when multiple data types were used to order and orient sequences. Perhaps more importantly, making changes to an assembly represented this way is much simpler. For example, to change the orientation of a contig in a fasta file of scaffolds, it is necessary to extract the sequence of this contig from its scaffold, reverse-complement the sequence, and then splice it back into the original file; however, performing the same operation in an AGP file is as simple as changing a "+" to a "−" on a single line.

Despite the wide use of AGP format among different genome assembly software, and the necessity to make edits to assemblies during assembly curation, there is not currently a published software package providing a simple way to make changes to an AGP file. Instead, those assembling genomes using multiple data sources must either write custom scripts, manually edit the AGP file, or forego the advantages of representing assemblies in AGP format and perform all steps in fasta format. Therefore, we developed agptools, a suite of command-line utilities that provides a simple interface for manipulating AGP files. This package also exposes an API that other developers can use to read, write, and manipulate these files in their own software, rather than needing to write their own code for these tasks.

2 Availability

Agptools is available under the open-source MIT License. It can be installed on any operating system running Python with one command. Most sub-commands can be run on a personal computer and require <1 GB RAM. We commit to maintaining the code and documentation for two years after publication on GitHub at https://github.com/WarrenLab/agptools. The version described in this manuscript will be available permanently via Zenodo with DOI 10.5281/zenodo.15085069.

3 Implementation

Agptools is written in Python and can be run on Python versions ≥ 3.7. It uses the screed (scr,) and pyfaidx (Shirley *et al.* 2015) packages for fasta input and output. Its command-line interface consists of a single base command, "agptools", followed by a sub-command. Most subcommands take an AGP file and a second text file listing operations to perform as input, and output a modified AGP file. The subcommands are described below.

3.1 Join

The join operation takes two or more input sequences and joins them into a single output sequence, delimited by gaps (Fig. 1a). It takes as input an AGP file and a list of joins to make, and outputs an AGP file with the requested sequences joined. The list of joins is a file where each line contains a list of input sequences to join into a single output sequence, along with their order and orientation. The size, type, and evidence listed on the newly created gap lines can be specified by optional arguments.

3.2 Split

The split operation takes a single sequence and breaks it into two or more sequences at specified breakpoints (Fig. 1b). It takes as input an AGP file and a list of breakpoints, and outputs an AGP with the input sequences split at the given breakpoints. The list of breakpoints is a tab-separated file with two fields: the first is the name of the sequence to split and the second is a comma-separated list of breakpoints within that sequence. Sequences can be split both within gaps and within non-gap sequence.

3.3 Flip

The flip operation inverts part of a sequence (Fig. 1c). It takes as input an AGP file and a list of segments to flip, and outputs an AGP file with the requested segments in the opposite orientation. The list of segments to invert is a tab-separated file with three columns: sequence name, start coordinate of segment to invert, and end coordinate of segment to invert.

3.4 Remove

The remove subcommand removes sequences from the input. This can be useful when curating an assembly to remove haplotigs or sequences representing contamination. It takes as input an AGP file and a list of names of sequences to remove, and outputs an AGP without the removed sequences.

3.5 Rename

The rename subcommand changes the names of input sequences. This can be useful for converting an assembly from generic computer-generated sequence names to chromosome or plasmid names. It takes as input an AGP file along with a tab-separated file with three columns: old sequence name, new sequence name, and new sequence orientation, and outputs an AGP file with sequences renamed.

3.6 Assemble

The assemble subcommand takes a fasta containing input sequences and an AGP file describing how these sequences are laid out into larger sequences, and outputs a fasta file with the input sequences laid out as described in the AGP (Fig. 1d). This can be helpful for performing whole-genome alignment to use synteny with a related species for curation. Because whole contigs are loaded into memory in this command, it requires more RAM than the other subcommands.

3.7 Transform

Making changes to an assembly presents the problem of introducing new coordinate systems. For example, if reads were aligned to contigs, the coordinates of these alignments will not be the same once the contigs are assembled into scaffolds. The transform subcommand overcomes this problem by translating coordinates from one system to another (Fig. 1e). As input, it takes a bed file with coordinates relative to the input sequences and an AGP that assembles input sequences into output sequences, and outputs a bed file with coordinates relative to the output sequences. The transform command is especially useful for translating bed coordinates of Hi-C alignments from contig coordinates to scaffold coordinates, so that heatmaps can be generated iteratively during curation without needing to realign Hi-C reads.

3.8 Sanitize

There are some constructs that are valid AGP format but not accepted by NCBI's assembly submission pipeline, such as single-component sequences with reverse-orientation and input contigs split among multiple output scaffolds. The

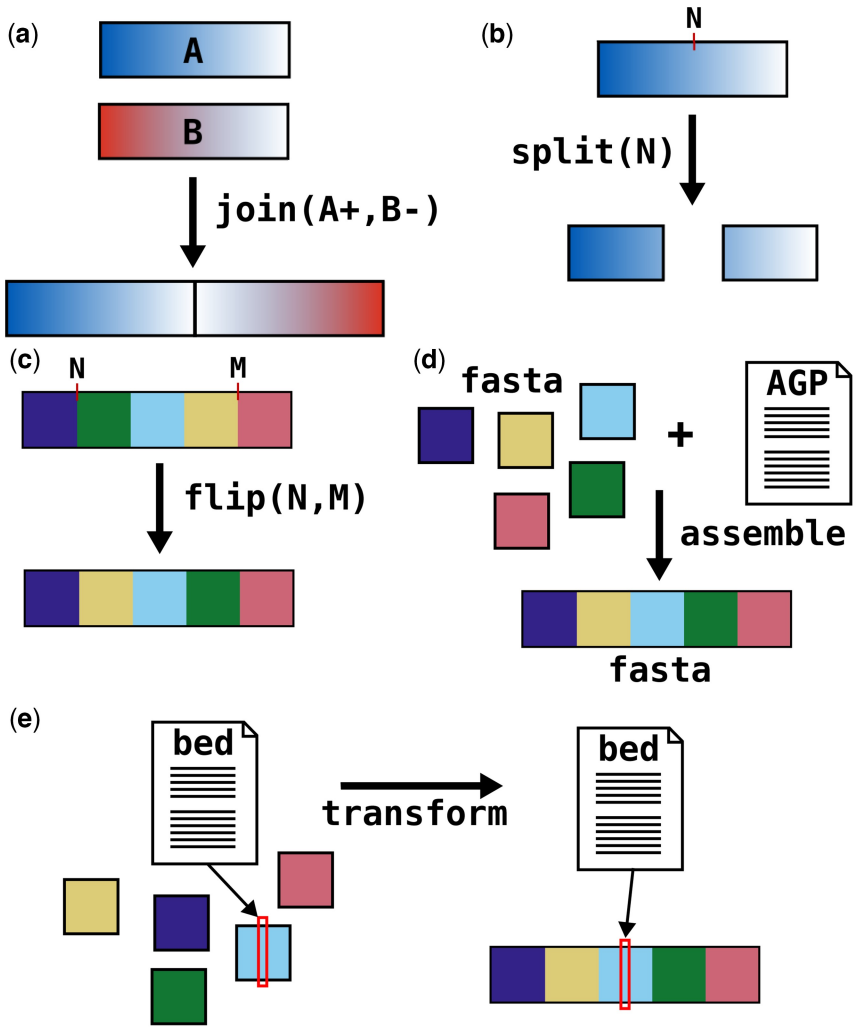

Figure 1. Subcommands of agptools. (a) The join command joins two or more separate sequences into a single sequence. (b) The split command splits a sequence at a given location into two sequences. (c) The flip command inverts part of a sequence. (d) The assemble command produces an assembled fasta file from a fasta file of sequence components and an AGP file giving their order and orientation. (e) The transform command translates coordinates in a bed file into the coordinate system of assembled sequences.

sanitize subcommand fixes these issues so that they do not cause errors during submission.

3.9 API

All these operations, in addition to being usable via a command-line interface, can also be performed within external python code using the API. This API also exposes functions and data structures for input and output of files in AGP format. Therefore, the software can be used by other developers to robustly manipulate AGP files within their own programs.

3.10 Documentation

The documentation for this software package takes three forms. First, command-line help messages for the root command as well as each subcommand explain how to use the command-line interface, including input, output, and additional arguments where relevant. Second, a more comprehensive description of each command is hosted on github, with example input, command-line invocation, and output, as well as any additional notes on how it should be used. Finally, all classes and functions in the API are comprehensively documented in an API page.

3.11 Testing

Unit tests provide 100% code coverage of this software package using the pytest framework. All pull requests must pass all unit tests to be merged into the main branch of the repository.

3.12 Error handling

Incorrectly formatted input has the potential to cause a program to either crash with an error message that does not point to the source of the problem, or, worse, to cause undefined behavior, providing incorrect output with no warning or error. To mitigate this, we included error handling to catch many kinds of incorrectly formatted or nonsensical input and exit the program with an error message pointing to the input file causing the problem, and when possible, also the line number of the first problematic line. We also include unit tests that give bad input to the program and only pass if the program then raises the correct error.

4 Discussion

This software package provides a simple way for a user to manipulate genome assemblies during manual curation steps. It has already been used during the curation of several published genome assemblies (Carroll *et al.* 2024, Warren *et al.* 2024).

Agptools is written in Python, perhaps the most common programming language for bioinformatics, with MyPy type checking, linting, and autoformatting, and thorough source-code documentation. Moreover, numerous unit tests provide full code coverage of the package and decrease the likelihood that updates and bug fixes will introduce new bugs that break existing functionality. These factors, along with the simplicity of the software, make it highly maintainable.

One additional benefit of this software is that because all operations are performed as a single command with a file listing changes to make, assemblies curated using this package are highly reproducible. To reproduce the curation exactly, only the list of commands and the input files are necessary. This is different from curation software such as Juicebox Assembly Tools (Dudchenko *et al.* 2018), where edits are made by clicking on a heatmap, and therefore, two people making the same series of edits are likely to produce slightly different results by clicking in slightly different places.

Acknowledgements

We thank all users of this software who have found and reported bugs.

Author contributions

Edward Ricemeyer (Conceptualization [lead], Investigation [lead], Methodology [lead], Software [lead], Validation [lead], Writing—original draft [lead]), Rachel Carroll (Software [supporting], Writing—review & editing [equal]), and Wes Warren (Conceptualization [supporting], Project administration [lead], Writing—review & editing [equal])

Conflict of interest: None declared.

Funding

National Science Foundation collaborative research NSF DBI 1940624.

Data availability

No new data were generated or analysed in support of this research.

References

AGP Specification v2.1. screed: a simple read-only sequence database, designed for short reads. https://www.ncbi.nlm.nih.gov/genbank/genome_agp_specification/ (3 July 2025, date last accessed).

Alonge M, Lebeigle L, Kirsche M *et al.* Automated assembly scaffolding using RagTag elevates a new tomato system for high-throughput genome editing. *Genome Biol* 2022;**23**:258.

Belser C, Baurens F-C, Noel B *et al.* Telomere-to-telomere gapless chromosomes of banana using nanopore sequencing. *Commun Biol* 2021;**4**:1047.

Bliznina A, Masunaga A, Mansfield MJ *et al.* Telomere-to-telomere assembly of the genome of an individual oikopleura dioica from Okinawa using nanopore-based sequencing. *BMC Genomics* 2021;**22**:222.

Carroll RA, Rice ES, Murphy WJ *et al.* A chromosome-scale fishing cat reference genome for the evaluation of potential germline risk variants. *Sci Rep* 2024;**14**:8073.

Cheng H, Asri M, Lucas J *et al.* Scalable telomere-to-telomere assembly for diploid and polyploid genomes with double graph. *Nat Methods* 2024;**21**:967–70.

Cheng H, Jarvis ED, Fedrigo O *et al.* Haplotype-resolved assembly of diploid genomes without parental data. *Nat Biotechnol* 2022;**40**:1332–5.

Dudchenko O, Shamim MS, Batra S *et al.* The juicebox assembly tools module facilitates de novo assembly of mammalian genomes with chromosome-length scaffolds for under $1000. bioRxiv, 2018, preprint: not peer reviewed.

Ebler J, Ebert P, Clarke WE *et al.* Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes. *Nat Genet* 2022;**54**:518–25.

Ghurye J, Rhie A, Walenz BP *et al.* Integrating hi-C links with assembly graphs for chromosome-scale assembly. *PLoS Comput Biol* 2019;**15**:e1007273.

Groza C, Schwendinger-Schreck C, Cheung WA *et al.* Pangenome graphs improve the analysis of structural variants in rare genetic diseases. *Nat Commun* 2024;**15**:657.

Jain M, Koren S, Miga KH *et al.* Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nat Biotechnol* 2018;**36**:338–45.

Leonard AS, Crysnanto D, Mapel XM *et al.* Graph construction method impacts variation representation and analyses in a bovine super-pangenome. *Genome Biol* 2023;**24**:124.

Nurk S, Koren S, Rhie A *et al.* The complete sequence of a human genome. *Science* 2022;**376**:44–53.

Olagunju TA, Rosen BD, Neibergs HL *et al.* Telomere-to-telomere assemblies of cattle and sheep Y-chromosomes uncover divergent structure and gene content. *Nat Commun* 2024;**15**:8277.

Rautiainen M, Nurk S, Walenz BP *et al.* Telomere-to-telomere assembly of diploid chromosomes with verkko. *Nat Biotechnol* 2023;**41**:1474–82.

Rhie A, McCarthy SA, Fedrigo O, 3rd, *et al.* Towards complete and error-free genome assemblies of all vertebrate species. *Nature* 2021;**592**:737–46.

Rice ES, Green RE. New approaches for genome assembly and scaffolding. *Annu Rev Anim Biosci* 2019;**7**:17–40.

Shirley MD, Ma Z, Pedersen BS *et al.* Efficient "pythonic" access to FASTA files using pyfaidx. *PeerJ* 2015;**3**:e970v1.

Smith TPL, Bickhart DM, Boichard D *et al.* Bovine Pangenome Consortium. The bovine pangenome consortium: democratizing production and accessibility of genome assemblies for global cattle breeds and other bovine species. *Genome Biol* 2023;**24**:139.

Warren WC, Rice ES, X M *et al.* Astyanax mexicanus surface and cavefish chromosome-scale assemblies for trait variation discovery. *G3 (Bethesda)* 2024;**14**:jkae103.

Xue L, Gao Y, Wu M *et al.* Telomere-to-telomere assembly of a fish Y chromosome reveals the origin of a young sex chromosome pair. *Genome Biol* 2021;**22**:203.

Zhou C, McCarthy SA, Durbin R. YaHS: yet another hi-C scaffolding tool. *Bioinformatics* 2023;**39**:btac808.

Bioinformatics, 2025, 41, 1–4
https://doi.org/10.1093/bioinformatics/btaf388
Applications Note