GraTools, an user-friendly tool for exploring and manipulating pangenome variation graphs

Sebastien Ravel[1*†], Nina Marthe[2†], Camille Carrette[2,3†], Mourdas Mohamed[2], Francois Sabot[2*], Christine Tranchant-Dubreuil[2*]

[1*]UMR PHIM, Plant Health Institute of Montpellier, Univ Montpellier, INRAe, CIRAD, Institut Agro, Campus International de Baillarguet, Bâtiment K, Montpellier Cedex 5, 34398, France.
[2]DIADE Unit, University of Montpellier, Cirad, IRD, 911 Avenue Agropolis BP64502, Montpellier, 34394, Occitanie, France.
[3]Crops Bioinformatics Team, Syngenta Seeds SAS, 1228 Chemin de l'Hobit, Saint-Sauveur, 31790, Occitanie, France.

*Corresponding author(s). E-mail(s): sebastien.ravel@cirad.fr; francois.sabot@ird.fr; christine.tranchant@ird.fr;
Contributing authors: nina.marthe@ird.fr; camille.carrette@ird.fr; mourdas.mohamed@ird.fr;
†These authors are co-first authors.

Abstract

Pangenome variation graphs (PVG), which represent genomic diversity through multiple genomes alignment, are powerful tools for studying genomic variations in populations. However, current tools often lack integration, efficiency, or require format conversions, to use them, hindering their usability. Here, we introduce GraTools, a fast and user-friendly command-line tool for manipulating PVG in their native GFA format. GraTools indexes PVG once, enabling rapid subgraph extraction, fasta sequence recovery, and comprehensive analyses — including core/dispensable genome ratios or group-specific segment identification. Its modular architecture relies on optimized data structures (BAM/BED files), parallel processing, and asynchronous I/O for scalability. GraTools supports also non-reference coordinates and integrates with visualization tools like Bandage. Using an Asian rice pangenome graph (13 accessions), we demonstrate its ability to extract subgraphs, compute depth statistics, and identify subspecies-specific segments. The tool intuitive interface, its real-time feedback and its detailed logging

allows its usability for diverse applications, from population genetics to breeding and genomic medicine, from biologists to bioinformaticians. Through a unified graph manipulation interface, GraTools overcomes the limitations of fragmented workflows for PVG use. It is available as an open-source tools (GNU GPLv3), with its documentation available at https://gratools.readthedocs.io.

Keywords: Pangenome graph, subgraph, core genome

1 Introduction

Nowadays, more and more studies on diversity, agronomy, health or evolution, at the species or genera level, focus on the concept of pangenome. This approach, considering the whole genomic content of a group of individuals [1, 2] and not of a single individual, allows to better identify variations and thus the evolution and functional role of sequences (e.g. Sub1A gene in rice, [3]). While many type of pangenome approaches exist [1, 4], in higher eukaryotes and for species-level analyses, most of the studies focus on pangenome variation graph for representing the pangenomic sequence variations. These graphs are directed acyclic graph, where nodes (or segments) represent sequences of different sizes, and edges (or links) an adjacency between two sequences within at least one of the haplotypes (or path) embedded in the graph [5, 6]. The current format for storing pangenome variation graphs is the GFAv1.1 (Graphical Fragment Assembly [7]), generally built from a MSA-like approach [8], [9]. This text-based format is structured in S, L and W/P lines, for segments, links and walks/paths, respectively. Using the W/P lines, one can reconstruct the sequence of an embedded individual through the sequences of the nodes within the S lines. Many tools propose a GFA output for such variation graph, such as MiniGraph/Cactus [9], PGGB [8], vg [5], etc... In addition, different tools propose also some basic statistics (gfastats [10], pancat [11]), basic manipulation (vg [5], odgi [12], gfatools [13]), extracting subgraph/partial graph (pancat [11]), or extracting some individual sequences (vg [5], odgi [12]). However, none of those tools provide an integrated suite of tools for graph manipulation or an intuitive interface, some suffer main withdraws in terms of memory or CPU time usage or both, or have to transform the initial GFA graph in their own format or renaming nodes.

Here, we propose GraTools, an intuitive and fast command-line tool for manipulating pangenome variation graphs under the native GFA format and node names. It indexes the graph once and allows users to extract subgraphs and fasta sequences either from a given location (genome path or graph nodes) or in composition, provides statistics on the graph and the Core/Dispensable compartment, as well as analyses on the graph topology. It is primarily designed to work with GFA coming from MiniGraph/Cactus [9] but it also works with ones coming from other tools such as PGGB [8]. GraTools is developed in Python, installable using *pip* and an *Apptainer* container. The code being available under the GNU GPLv3 license at https://forge.ird.fr/diade/gratools, and its complete manual at https://gratools.readthedocs.io/.

2 Implementation

GraTools is implemented in Python 3.10+ and designed to be a robust and efficient tool for pangenomic variation graph analysis. It relies on a minimal set of external dependencies, the most notable being the BEDtools [14] command-line suite, whose must be accessible in the system executable path.

2.1 Installation and manual

The tool is distributed *via* the Python Package Index (PyPI) for straightforward installation and can also be installed from its source code repository or through an *Apptainer* container. Comprehensive documentation, including installation guides, tutorials, and a full command reference, is maintained on *ReadTheDocs* (https://gratools.readthedocs.io).

2.2 One-Time Indexing

GraTools performance model is built upon a one-time indexing process. During the initial run on a GFA file, the graph structure is parsed and recorded into highly optimized data formats: an indexed BAM file for segment data, whose manipulation is powered by the *pysam* API [15–18], and a collection of BED files mapping the segment coordinates for each of the genomic path. This indexing step is the most time-intensive part of the workflow, typically lasting from 2 to 30 minutes depending on the graph's scale, but the resulting index is stored persistently. Thus, all subsequent analyses and queries will reuse this index, allowing for a drastic drop-down of the execution times.

2.3 Architecture and Core Components

`GraTools` is built on a modular, class-based architecture where each component has a clearly defined role. A central `Gratools` class coordinates the entire workflow, as illustrated in Figure 1.

Gratools Class

This class acts as the main orchestrator of the workflow. It serves as the bridge between the command-line interface and the backend processing modules. This class interprets the user commands, manages input and output paths, and coordinates the execution of the analysis pipelines.

GFA Class

This class is dedicated to the initial parsing and processing of the input GFA file. It systematically reads and interprets each line type (Header, Segment, Link, Walk), extracting critical information and populating the primary data structures. During this phase, it builds a dictionary of all segments and their lengths, and dispatches the path information to the `AsyncBedWriter` for BED file creation and the link information to the `AsyncGfaDatabase` for database indexing (optional).

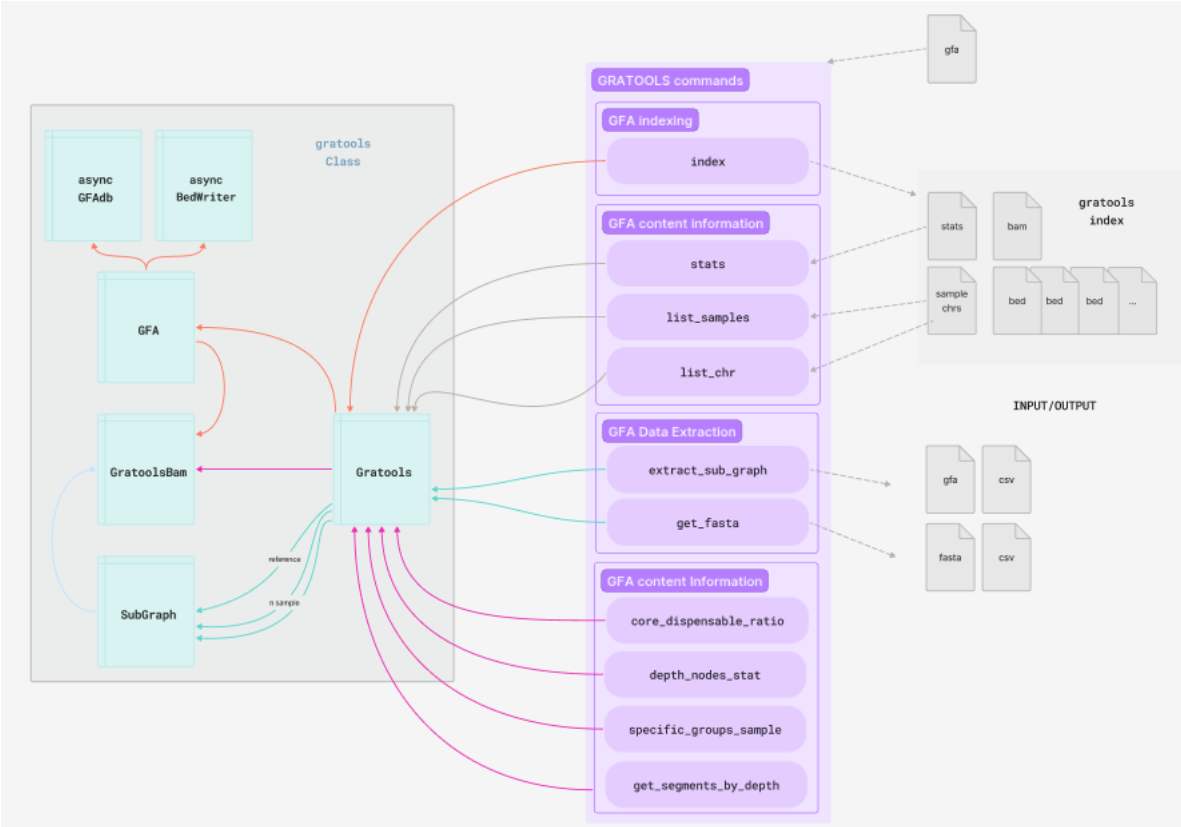

Fig. 1 Overview of `GraTools`. The central `Gratools` class coordinates the interactions with three core classes: `GFA`, `GratoolsBam`, and `SubGraph`. The functionalities and commands are organized into four categories: **GFA indexing** (`index`), **content information** (`stats`, `list_samples`, `list_chr`), **data extraction** (`extract_sub_graph`, `get_fasta`), and **content analysis** (`core_dispensable_ratio`, `depth_nodes_stat`, `specific_groups_sample`, `get_segments_by_depth`). The GFA files serve as input, and outputs are provided in BED, BAM, FASTA, GFA and CSV formats, enabling a comprehensive analysis of pangenome variation graphs.

GratoolsBam Class

The `GratoolsBam` class encapsulates all interactions with the indexed BAM file created during the initial parsing. It handles essential operations such as indexing this BAM file, extracting specific segments, and adding or modifying segment tags (e.g., the custom 'SW' tag that lists which samples contain a given segment). This class serves as the computational engine for most of the analytical commands, performing the calculations for the core/dispensable ratios, segment depth statistics, or the identification of shared segments between sample groups, by efficiently iterating through the indexed BAM records.

SubGraph Class

The `SubGraph` class is designed to extract and process a specific region of interest within the larger pangenome graph. When a user requests a subgraph for a given genomic coordinate range (from any embedded genome), this class is instantiated. It uses ***pybedtools*** [19] to intersect the query coordinates with the pre-computed BED files, identifying all relevant segments and paths, and reconstructing the corresponding subgraph in memory. From this in-memory representation, it can generate outputs in graph GFA format (with native node names), or linear FASTA format (through the ***Biopython*** library [20]).

AsyncGfaDatabase and AsyncBedWriter

These specialized worker classes are responsible for handling I/O operations during the indexing phase. The `AsyncBedWriter` manages the creation of all the BED files for individual paths. As the `GFA` class parses the walk lines, it sends formatted BED entries to a queue. The writer runs in a separate, non-blocking task, consuming from this queue and writing data to the appropriate per-sample BED files on disk. The `AsyncGfaDatabase` class manages an asynchronous SQLite database to store and query all link data from the GFA file (optional). It uses batch insertion to efficiently store the millions of links during the initial parse. Once populated, it creates indexes on segment ID columns to enable high-speed queries for graph traversal and connectivity analysis, such as finding all segments connected to a given node, or identifying distinct connected components within the graph.

2.4 Performance and Scalability

To ensure efficient processing of large-scale pangenome graphs, GraTools was engineered with performance as a primary consideration.

Parallel Processing

CPU-bound tasks, particularly those involving iterating over many samples, are parallelized using the Python `concurrent.futures.ProcessPoolExecutor` [21]. This strategy allows GraTools to fully utilize multi-core systems (defined by user) by distributing work across multiple processes, drastically reducing computational time for large analyses.

Asynchronous I/O

I/O-intensive operations during the indexing are managed asynchronously to prevent disk writing to become a main bottleneck. GraTools uses the Python `asyncio` framework with the high-performance `uvloop` event loop [22]. The `AsyncGfaDatabase` class is powered by the specialized library ***aiosqlite*** [23] for database interactions, while the `AsyncBedWriter` class uses the ***aiofiles*** [24] library for file writing. These classes operate as non-blocking background tasks, receiving data through queues, which allows the main parsing process to continue without waiting for disk I/O to complete.

Efficient Data Handling

Post-indexing, all data accesses are mediated through the compressed and indexed BAM/BED files, avoiding the need for repeated parsing of the large text-based GFA file. Data manipulation is handled by the ***pandas*** [25, 26] and ***NumPy*** [27] computational libraries, and memory usage is kept low by streaming data when possible.

2.5 User Experience and Interactivity

A core design goal was to create a command-line tool that is efficient, intuitive for most users, and transparent in its operations.

Command-Line Interface

The CLI is built with a combination of the ***Click***, ***click-extra***, and ***cloup*** libraries [28–30], in order to provide a clean, structured, and color-coded user interface. Commands are organized into logical groups, and detailed helps are available for every commands and options. For an enhanced usability, a shell completion command is provided to facilitate rapid and accurate command entry.

Interactive Feedback

For long-running operations like indexing or parallel analysis, GraTools provides detailed, real-time visual feedback using the ***rich*** library [31]. Multi-level progress bars track both the overall progress across all tasks and the status of individual sub-tasks (e.g., per-sample processing), offering a complete transparency of the tool execution.

Advanced Logging

GraTools incorporates a custom, non-blocking logging handler system. The `ThreadedRichHandler` processes log records in a separate background thread, ensuring that logging activity does not impact the performance of the main application. The logger supports multiple verbosity levels and produces a richly formatted, color-coded output for enhanced readability (terminal level).

3 Results

GraTools can perform various operations on the graph contained in the provided GFA file (as described in Table 1), that previously required many different tools to be used. The GraTools features can be organized in three main sections: basic description of the graph, extraction of subgraphs and of fasta sequences, and analyses of the graph content and of its genomic diversity. For simplicity of the description latter on, any haplotype or genome contained in the graph will be referred as a path. The examples of usage presented here are based on the Asian Rice graph with 13 accessions/haplotypes provided in [32, 33], but the tool has been tested in a large variety of pangenome variation graphs from different species.

	Gfatools	Pancat	ODGI	VG	Gratools
GFA input	X	X	X	X	X
GFA output (native node names)					X
Extract stat	X	X	X	X	X
Extract subgraph		X	X	X	X
Extract fasta		X		X	X
Extract segments			X		X
Using non-reference coordinate system					X
Bandage output					X
Visualization			X		
Index			X	X	X

Table 1 Comparative table of Gratools and other manipulation tools

3.1 Description of the graph content

3.1.1 Obtaining basic statistics and indexing

Similarly to many tools working on GFA, GraTools proposes basic statistics on the graph itself with the *stats* command: number and mean/minimal/maximal size of nodes, mean and maximum number of output branching, total graph size, etc... An example of the outputted statistics is available in SuppData. These statistics are performed during the indexing; however they can be called separately from the *index* command (that is performed only once, as described in the 2.2 part) latter on.

3.1.2 Listing the path names and the chromosome groups

Through the index analysis, the *list_indiv* command will output the list of individual paths embedded in the graph, while the *list_chromosomes* output the chromosomes (or contigs) list organized by group. These features can be helpful to quickly explore a graph, to know which samples were embedded in it, or what are the homologous chromosomes/contigs that the graph builder considered (if any).

3.2 Extraction of parts of the graph

3.2.1 Extracting a subgraph

The *extract_subgraph* will, as it names indicates, extracts a subgraph based on the coordinates of *any* of the embedded path, not only based on the reference coordinates. It allows then to visualize the subgraph of a complex region in an easier way than by opening the whole graph, using Bandage [34], e.g., as shown in Figure **??**. The extracted subgraph contains by default the whole set of path, and it can be restricted to an user-defined list of these. The node names in the newly extracted subgraph GFA file are the same as in the original graph, for a better forward analysis. In addition to the GFA file itself, GraTools can also output the corresponding fasta sequences for each (required) path.

3.2.2 Recovering fasta sequences from a subgraph/subregion

The *get_fasta* command works quite as the previous one, but provides only the fasta sequences for all the embedded paths corresponding to the coordinates of one of the embedded path. As previously, it can be restricted to a list of paths only. In addition, the provided coordinates can be from a path that will not be included in the restricted output.

3.3 Analyses on the graph

3.3.1 Obtaining the depth of nodes

The *depth_node_stat* option will deliver a tabular file containing per line the number of nodes traversed by n paths, then by n-1 and so on, n being the maximum number of embedded paths. The minimum number of paths to consider can be restricted, in order to avoid the individual sequences.

3.3.2 Extracting nodes using a depth threshold

The *get_segment_by_depth* option will recover a list of segments that are shared by a certain number of paths within the graph. Upper and lower bounds must be specified, as integer or percentage of samples, to allow the recovery of the node list. Users can for instance ask the recovery of the nodes shared by a maximum of 50% of the embedded paths but with a minimum of 25% of them too. As shown in Figure 2, the output includes a CSV file that can be incorporated in Bandage [34] to color the node with the specified depth.

3.3.3 Core/Dispensable genome ratio

The *core_dispensable_ratio* function proposes to users to extract the ratio of Core *vs.* dispensable genome, in fixing themselves the percentage of sharing requested for a node being categorized as Core. Indeed, one can specify a strict core (100% of sharing, 0% of missing), or a more relaxed one (90% of sharing e.g.). This option helps to identify common regions.

3.3.4 Recovering segments based on group specificity

The *specific_groups_sample* option proposes to extract nodes that are specific from a set of paths compared to another, or shared by a set, or absent in it. It can be used for instance to recover nodes specific to a population compared to another (wild and cultivated, e.g.), or to find the nodes that are common to the whole population of a subspecies (or absent in it).

The Figure 3 shown an application in deciphering the number of nodes and their cumulated size that are specific from each of the rice subspecies in the rice graph used in [32]. Once the node list obtained, we simply sum their size using the pysam API upon the BAM file containing the nodes information (see SuppData for the detailed commands). As expected, in this analyses, the *indica* (8 samples) specific part is larger than the *japonica* (5 samples) one, as shown in [33]. In Asian rice, the expecting

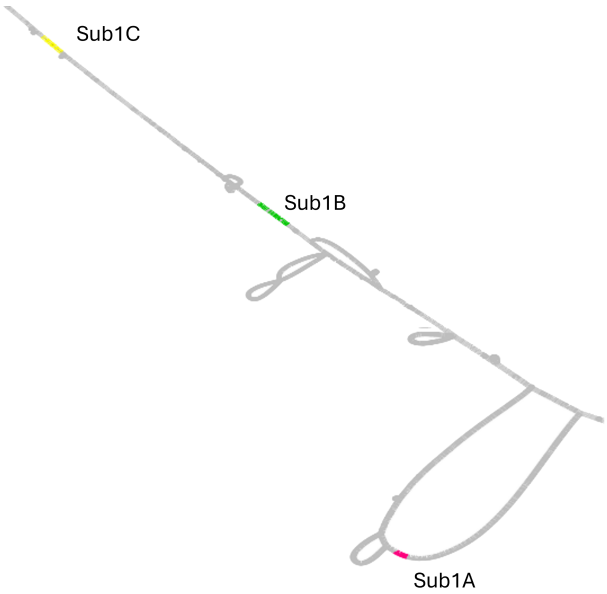

Fig. 2 Bandage representation of the Sub1 locus in Asian rice graph from [32].The 3 genes of the locus, namely Sub1C, Sub 1B and Sub1A are depicted in yellow, green and red, respectively on the graph structure.

genome size if of 380 to 410Mb, and here the core part represent roughly 90-95% of the genome of each individual. The mean size for each node in the core part is of 44bp, while it is of 249 and 308 bp for *indica* and *japonica* part, showing that most of the variations between the two groups are not SNP but larger variations, as shown in [35].

4 Conclusion

In the present paper, we present GraTools, an intuitive and user-friendly command line tool to manipulate, extract and analyze pangenome variation graphs in GFA format. It has been tested on graphs from various origins and species, and has been shown to be fast and efficient. It was designed with in mind the use of GFA for subsequent analyses, in particular population genetics or evolution ones, but can also be used for breeding (detection of specific nodes in a set of elites lines e.g.), genomic medicine (detection of missing/additional nodes in population with genetic illness) or even antibiotic resistance analyses (detection of additional segments in resistant strains). As a large set of commands are included within GraTools, users can manipulate their GFA within a single environment, and re-use it fastly due to the conservation of the intermediate files and indexes in their user space (or shared one). The outputs, messages and log files are human-readable and understandable, allowing users to manage easily them. Thus, GraTools streamlines the manipulation of pangenome variation graph in GFA format, allowing to really unleash the potential of pangenome graphs in higher eukaryots. We plan to add new features and options to GraTools in the near future, in particular for snarl extraction or evolution-related statistics.

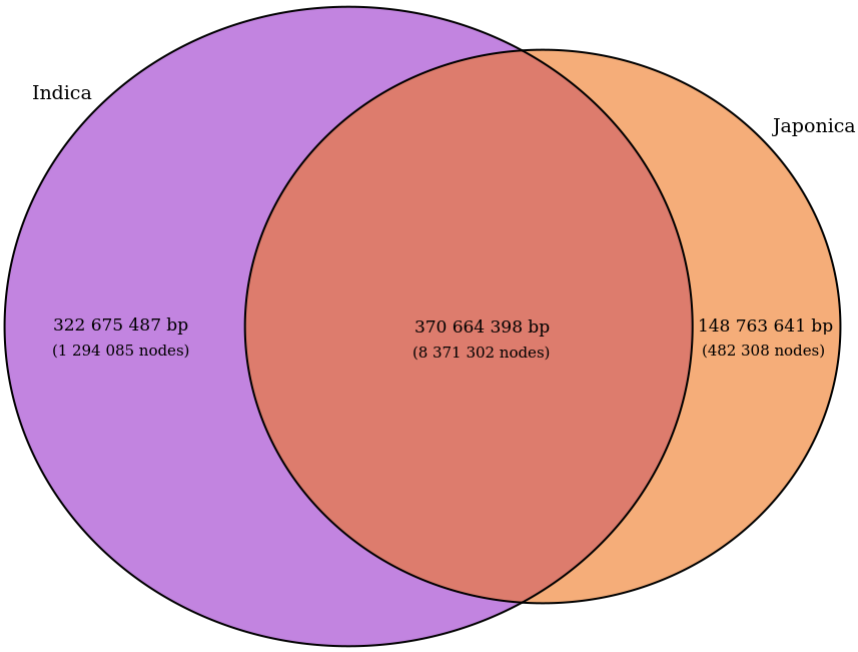

Fig. 3 Venn diagram representing the *indica* and *japonica* subspecies specific nodes in the Asian Rice graph from [32], computed using *specific_groups_samples*. For each subspecies (purple for the *indica*, and orange for the *japonica*), we indicates the sum in bp of the total nodes (specific or shared).

Supplementary information. Supplementary Table 1: example of stats

Acknowledgments. The authors acknowledge the ISO 9001 certified IRD i-Trop HPC at IRD Montpellier for providing HPC resources that have contributed to the research results reported within this paper. URL: https://bioinfo.ird.fr/. The analyses were also performed on the Core Cluster of the Institut Français de Bioinformatique (IFB) (ANR-11-INBS-0013)

Declarations

- Funding: Nina Marthe is supported by a PhD funding from the Agence Nationale de la Recherche as part of the France2030 program under the reference "ANR-22-PEAE-4". Camille Carrette is supported by a CIFRE ANRT between Syngenta Seeds SAS and IRD. Mourdas Mohamed is supported by a PlantAlliance/SofiProteol/IRD joint project.
- Conflict of interest/Competing interests: None
- Ethics approval and consent to participate: Not applicable

- Consent for publication: Yes
- Data availability: https://dataverse.ird.fr/diade/grannot
- Code availability https://forge.ird.fr/diade/gratools
- Author contribution: CTD, FS and SR managed the project; SR, NM, MM, CC and CTD implemented the code; SR, FS and all authors developed the algorithms; FS and SR drafted the manuscript; all authors corrected and validated the manuscript.

Appendix A Supplementary figures

GFA File: NewRiceGraph_MGC
Total samples: 13

--- GFA Statistics for: NewRiceGraph_MGC ---

Comprehensive GFA Statistics

Category	Metric	Value
Graph Overview	GFA File Name	NewRiceGraph_MGC
Graph Overview	GFA Version	1.1
Graph Overview	Total Segments (S lines)	26,461,214
Graph Overview	Total Links (L lines)	36,276,920
Graph Overview	Total Walks (W lines)	743
Graph Overview	Unique Samples in Walks	13
Segment Statistics	Total Segment Length (bp)	858,417,045
Segment Statistics	Average Segment Length (bp)	32.44
Segment Statistics	Median Segment Length (bp)	1.00
Segment Statistics	Avg Length of Top 5% Longest Segments (bp)	508.68
Segment Statistics	Median Length of Top 5% Longest Segments (bp)	155.00
Segment Statistics	Segment Length Distribution	0-499bp: 26325942, 500-999bp: 77342, 1000-1999bp: 28835, 2000+bp: 29095
Link Statistics	Max Segment Degree	0
Link Statistics	Average Segment Degree	2.74
Link Statistics	Self-Links (S1 -> S1)	0
Link Statistics	Inverted Links (S1+ -> S2-)	0
Link Statistics	Both Negative Links (S1- -> S2-)	0
Path (Walk) Statistics	Total Length of All Paths (bp, input_genome_size)	5,110,760,744
Path (Walk) Statistics	Graph Compression Ratio (input_genome_size / graph_size_bp)	5.95
Path (Walk) Statistics	Max Segments in a Single Walk	1,658,219
Path (Walk) Statistics	Sum of First Segment Lengths in Walks	5,442,916
Segment Sharing & Depth	Avg Unique Samples per Segment (Similarity Mean)	7.36
Segment Sharing & Depth	Median Unique Samples per Segment (Similarity Median)	7.00
Segment Sharing & Depth	StdDev Unique Samples per Segment (Similarity Std)	4.47
Segment Sharing & Depth	Avg Occurrences per Segment (Depth Mean)	7.49
Segment Sharing & Depth	Median Occurrences per Segment (Depth Median)	7.00
Segment Sharing & Depth	StdDev Occurrences per Segment (Depth Std)	4.59
Graph Structure	Graph Density	1.04e-07
Graph Structure	Segments/Links Ratio	0.73
Graph Structure	Dead-End Segments (degree 1)	0
Graph Structure	Isolated Segments (degree 0)	26,461,214
Graph Structure	Number of Connected Components (CCs)	0
Graph Structure	Largest CC Size (bp)	0
Graph Structure	Number of Disconnected CCs (excluding largest)	0
Graph Structure	Total Length of Disconnected CCs (bp)	0

Fig. A1 Example of statistics obtained using the **stats** command on the *NewRiceGraph_MGC.gfa.gz* graph: `gratools stats -g NewRiceGraph_MGC.gfa.gz`

```
Total segments analyzed: 26,461,214
Total segments passing length filter: 2,233,791 (8.44%)

        Node Depth Statistics — NewRiceGraph_MGC (Len ≥ 50bp)
```

Depth	Segments	Percentage	Filtered Segments	Filtered %
1	3,824,526	14.45%	40,656	1.82%
2	1,957,032	7.40%	111,129	4.97%
3	1,549,017	5.85%	97,450	4.36%
4	1,550,238	5.86%	91,515	4.10%
5	1,685,342	6.37%	98,449	4.41%
6	1,450,147	5.48%	83,489	3.74%
7	1,395,367	5.27%	79,614	3.56%
8	1,498,592	5.66%	84,251	3.77%
9	1,174,201	4.44%	55,274	2.47%
10	1,116,803	4.22%	53,726	2.41%
11	1,296,676	4.90%	57,559	2.58%
12	2,291,274	8.66%	110,461	4.95%
13	5,671,999	21.44%	1,270,218	56.86%

Fig. A2 Example of results obtained using the **depth_nodes_stat** command on the *NewRiceGraph_MGC.gfa.gz* graph for a minimum of 50%: `gratools depth_nodes_stat -g NewRiceGraph_MGC.gfa.gz -fl 50`

```
Total segments in GFA: 26,461,214
Total segments analyzed: 26,461,214
Total segments passing length filter (≥ 50bp): 2,233,791 (8.44%)
                    Core vs. Dispensable Segments — NewRiceGraph_MGC
```

Category	Count	Total	Percentage
Shared (Core) - Raw	7,963,273	26,461,214	30.09%
Specific (Dispensable) - Raw	3,824,526	26,461,214	14.45%
Shared (Core) - Filtered (Length >= 50bp)	1,380,679	2,233,791	61.81%
Specific (Dispensable) - Filtered (Length >= 50bp)	40,656	2,233,791	1.82%
Segments Filtered Out by Length	24,227,423	26,461,214	91.56%

Fig. A3 Example of results obtained using the **core_dispensable_ratio** command on the *NewRiceGraph_MGC.gfa.gz* graph for a ratio of 90% of core: `gratools core_dispensable_ratio -g NewRiceGraph_MGC.gfa.gz --input-as-percentage -sm 90 -spm 10 -fl 50`

References

[1] Matthews, C.A., Watson-Haigh, N.S., Burton, R.A., Sheppard, A.E.: A gentle introduction to pangenomics. Briefings in Bioinformatics **25**(6), 588 (2024) https://doi.org/10.1093/bib/bbae588 https://academic.oup.com/bib/article-pdf/25/6/bbae588/60702740/bbae588.pdf

[2] Tranchant-Dubreuil, C., Rouard, M., Sabot, F.: Plant Pangenome: Impacts on Phenotypes and Evolution, pp. 453–478. Wiley, ??? (2019). https://doi.org/10.1002/9781119312994.apr0664 . https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119312994.apr0664

[3] Xu, K., Xu, X., Fukao, T., Canlas, P., Maghirang-Rodriguez, R., Heuer, S., Ismail, A.M., Bailey-Serres, J., Ronald, P.C., Mackill, D.J.: Sub1a is an ethylene-response-factor-like gene that confers submergence tolerance to rice. Nature **442**(7103), 705–708 (206) https://doi.org/10.1038/nature04920

[4] Hu, Z., Wei, C., Li, Z.: In: Tettelin, H., Medini, D. (eds.) Computational Strategies for Eukaryotic Pangenome Analyses, pp. 293–307. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38281-0_13 . https://doi.org/10.1007/978-3-030-38281-0_13

[5] Garrison, E., Sirén, J., Novak, A.M., Hickey, G., Eizenga, J.M., Dawson, E.T., Jones, W., Garg, S., Markello, C., Lin, M.F., Paten, B., Durbin, R.: Variation graph toolkit improves read mapping by representing genetic variation in the reference. Nat. Biotechnol. **36**, 875–879 (2018) https://doi.org/10.1038/nbt.4227

[6] Liao, W.-W., Asri, M., Ebler, J., Doerr, D., Eizenga, J.M., Feng, X., Guarracino, A., Hickey, G., Horan, K., *et al.*: A draft human pangenome reference. Nature **617**(7960), 312–324 (2023)

[7] Graphical Fragment Assembly (GFA) Specification. https://gfa-spec.github.io/GFA-spec/GFA1.html (2024)

[8] Garrison, E., Guarracino, A., Heumos, S., Villani, F., Bao, Z., Tattini, L., Hagmann, J., Vorbrugg, S., Marco-Sola, S., Kubica, C., Ashbrook, D.G., Thorell, K., Rusholme-Pilcher, R.L., Liti, G., Rudbeck, E., Golicz, A.A., Nahnsen, S., Yang, Z., Mwaniki, M.N., Nobrega, F.L., Wu, Y., Chen, H., Ligt, J., Sudmant, P.H., Huang, S., Weigel, D., Soranzo, N., Colonna, V., Williams, R.W., Prins, P.: Building pangenome graphs. Nature Methods 2024 21:11 **21**, 2008–2012 (2024) https://doi.org/10.1038/s41592-024-02430-3

[9] Hickey, G., Monlong, J., Ebler, J., Novak, A.M., Eizenga, J.M., Gao, Y., Zook, J.M., Zimin, A.V., Wood, J.M.D., Wang, T., Walenz, B., Wagner, J., Vollger, M.R., Villani, F., Tricomi, F.F., Tomlinson, C., Thibaud-Nissen, F., Tayoun, A.N.A., Sofia, H.J., Smith, M.W., Sirén, J., Sibbesen, J.A., Shafin, K., Schultz, B.I., Schneider, V.A., Sanders, A.D., Sacco, S., Rhie, A., Regier, A.A., Rautiainen, M., Puiu, D., Prins, P., Porubsky, D., Popejoy, A.B., Phillippy, A.M., Pesout, T., Olson, N.D., Olsen, H.E., Nurk, S., Nattestad, M., Mwaniki, M.N., Munson, K.M., Mountcastle, J., Mitchell, M.W., Miga, K.H., McDaniel, J., McCartney, A., Martin, F.J., Marschall, T., Markello, C., Marijon, P., Marco-Sola, S., Magalhães, H., Lucas, J.K., Lu, T.Y., Lu, S., Liao, W.W., Lewis, A.P., Lee, H.J., Koren, S., Kordosky, J., Korbel, J.O., Kolesnikov, A., Koenig, B.A., Kenny, E.E., Ji, H.P., Jarvis, E.D., Jain, M., Howe, K., Hourlier, T., Hoekzema, K., Heumos, S., Haussler, D., Haukness, M., Harvey, W.T., Hall, I.M., Haggerty, L., Guarracino, A., Groza, C., Green, R.E., Giron, C.G., Garrison, N.A., Garrison, E., Garg, S., Fulton, R.S., Frankish, A., Formenti, G., Flicek, P., Fischer, C., Feng, X., Felsenfeld, A.L., Fedrigo, O., Fairley, S., Eichler, E.E., Ebert, P., Doerr, D., Diekhans, M., Cornejo, O.E., Cook-Deegan, R.M., Cook, D.E., Colonna, V., Cody, S., Chu, J., Cheng, H., Chang, X.H., Chang, P.C., Chaisson, M.J.P., Carroll, A., Buonaiuto, S., Bourque, G., Billis, K., Belyaeva, A., Baker, C.A., Baid, G., Asri, M., Antonacci-Fulton, L.L., Abel, H.J., Marschall, T., Li, H., Paten, B.: Pangenome graph construction from genome alignments with minigraph-cactus. Nature Biotechnology **42**, 663–673 (2024) https://doi.org/10.1038/S41587-023-01793-W;SUBJMETA=114,208,212,2785,631;KWRD=GENOME+INFORMATICS,GENOMICS

[10] Formenti, G., Abueg, L., Brajuka, A., Brajuka, N., Gallardo-Alba,

C., Giani, A., Fedrigo, O., Jarvis, E.D.: Gfastats: conversion, evaluation and manipulation of genome sequences using assembly graphs. Bioinformatics **38**(17), 4214–4216 (2022) https://doi.org/10.1093/bioinformatics/btac460 https://academic.oup.com/bioinformatics/article-pdf/38/17/4214/49889683/btac460.pdf

[11] Dubois, S., Zytnicki, M., Lemaitre, C., Faraut, T.: Pairwise graph edit distance characterizes the impact of the construction method on pangenome graphs. Bioinformatics **41** (2025) https://doi.org/10.1093/BIOINFORMATICS/BTAF291

[12] Guarracino, A., Heumos, S., Nahnsen, S., Prins, P., Garrison, E.: Odgi: Understanding pangenome graphs. Bioinformatics **38** (2022) https://doi.org/10.1093/bioinformatics/btac308

[13] Li, H.: gfatools: tools for manipulating sequence graphs in the GFA and rGFA formats. https://github.com/lh3/gfatools. GitHub repository (2024)

[14] Quinlan, A.R., Hall, I.M.: Bedtools: a flexible suite of utilities for comparing genomic features. Bioinformatics **26**(6), 841–842 (2010) https://doi.org/10.1093/bioinformatics/btq033

[15] Stepanov, V., contributors: pysam. https://github.com/pysam-developers/pysam. GitHub repository (2024)

[16] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., Subgroup, .G.P.D.P.: The sequence alignment/map format and samtools. Bioinformatics **25**(16), 2078–2079 (2009) https://doi.org/10.1093/bioinformatics/btp352

[17] Bonfield, J.K., Marshall, J., Danecek, P., Li, H., Ohan, V., Whitwham, A., Keane, T., Stalker, J.: Htslib: C library for reading/writing high-throughput sequencing data. GigaScience **10**(2), 007 (2021) https://doi.org/10.1093/gigascience/giab007

[18] Danecek, P., Bonfield, J.K., Liddle, J., Marshall, J., Ohan, V., Pollard, M.O., Whitwham, A., Keane, T., McCarthy, S.A., Davies, R.M., *et al.*: Twelve years of samtools and bcftools. GigaScience **10**(2), 008 (2021) https://doi.org/10.1093/gigascience/giab008

[19] Dale, R.K., Pedersen, B.S., Quinlan, A.R.: Pybedtools: a flexible python library for manipulating genomic interval data. Bioinformatics **27**(24), 3423–3424 (2011) https://doi.org/10.1093/bioinformatics/btr539

[20] Cock, P.J., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., *et al.*: Biopython: freely available python tools for computational molecular biology and bioinformatics. Bioinformatics **25**(11), 1422–1423 (2009)

[21] Foundation, P.S.: Python Language Reference, version 3.10 (2021). https://www.python.org

[22] Inc., M.: uvloop. https://github.com/MagicStack/uvloop (2016)

[23] Amis, J.: aiosqlite. https://github.com/omnilib/aiosqlite (2017)

[24] Tinto, M.: aiofiles. https://github.com/Tinche/aiofiles (2015)

[25] McKinney, W.: Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference, vol. 445, pp. 51–56 (2010). https://doi.org/10.25080/Majora-92bf1922-00a

[26] team, T.: pandas-dev/pandas: Pandas. Zenodo (2020). https://doi.org/10.5281/zenodo.3509134 . https://doi.org/10.5281/zenodo.3509134

[27] Harris, C.R., Millman, K.J., Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., *et al.*: Array programming with numpy. Nature **585**(7825), 357–362 (2020)

[28] Ronacher, A.: Click. https://github.com/pallets/click (2014)

[29] K-Jo: click-extra. https://github.com/k-jo/click-extra (2021)

[30] Ammar, G.: Cloup. https://github.com/janLuke/cloup (2021)

[31] Willison, T.: Rich. https://github.com/Textualize/rich (2020)

[32] Marthe, N., Zytnicki, M., Sabot, F.: Grannot, a tool for efficient and reliable annotation transfer through pangenome graph. Peer Community Journal **5** (2025) https://doi.org/10.24072/pcjournal.651

[33] Zhou, Y., Chebotarov, D., Kudrna, D., *et al.*: A platinum standard pan-genome resource that represents the population structure of asian rice. Scientific Data **7**, 113 (2020) https://doi.org/10.1038/s41597-020-0438-2

[34] Wick, R.R., Schultz, M.B., Zobel, J., Holt, K.E.: Bandage: interactive visualization of de novo genome assemblies. Bioinformatics **31**(20), 3350–3352 (2015) https://doi.org/10.1093/bioinformatics/btv383

[35] Qin, P., Lu, H., Du, H., Wang, H., Chen, W., Gao, Z., He, L., Li, L., Wang, W., Yu, H., *et al.*: Pan-genome analysis of 33 genetically diverse rice accessions reveals the genetic basis of subspecies differentiation and agronomic traits. Molecular Plant **14**(7), 1037–1053 (2021)