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

Background: Pangenome variation graphs (PVGs), 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.
Results: Here, we introduce GraTools, a fast and user-friendly command-line tool for manipulating PVGs using the original GFA file. After a one-time graph import, GraTools enables rapid subgraph extraction, fasta sequence retrieval, and comprehensive analyses, including core/dispensable genome ratio calculation or group-specific segment identification. The import step results in conversion in standard data formats (BAM/BED), enabling the reuse of well-optimized existing tools, allowing an efficient storage and the querying of the PVGs large complex data structures. Scalability is ensured by a modular architecture supporting parallel processing and asynchronous I/O operations. GraTools supports coordinates defined on both the primary reference as well as from alternative

genomes within the graph without re-importing, and its outputs can be easily visualized or manipulated using external tools. Using an Asian rice pangenome graph (13 accessions), we demonstrate its ability to easily extract subgraphs, compute depth statistics, and identify subspecies-specific segments. An intuitive command-line interface, a real-time execution feedback and a detailed logging system make this tool suitable for a wide range of applications, from population genetics to breeding and genomic medicine, for both biologists and bioinformaticians.

Conclusions: Through its unified graph manipulation interface, GraTools offers an interesting alternative to the few existing tools for manipulating PVGs, facilitating rapid, efficient and flexible downstream analyses. It is available as an open-source tool (GNU GPLv3), with its documentation available at https://gratools.readthedocs.io.

Keywords: Pangenome Variation Graph, Graph-based pangenome analysis, subgraph, core and dispensable genome, pangenome

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, that considers 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 recent studies focus on pangenome variation graphs (PVGs) for representing the pangenomic sequence variations. These PVGs, generally built from a MSA-like approach [5, 6], are directed graphs, in which nodes (or segments or fragments) represent sequences of varying lengths from at least one individual, and edges (or links) indicate adjacency between two sequences along at least one haplotype (or path or walk) in the PVG[7, 8]. Most of the currently used PVGs are acyclic ones, as recommended [9], even if a few of them are still partially cyclic ones.

The most currently used format for storing PVGs is the GFAv1.1 (Graphical Fragment Assembly [10]), a text tabular-based format structured in S, L and W/P lines, for segments, links and walks/paths, respectively. Using the W/P lines, one can reconstruct the genomic sequence of an embedded individual through the sequences of the nodes within the S lines. Other formats exist for storing PVGs, such as the rGFA or GFAv2.0, but are lacking some information (no P/W lines in rGFA e.g.), or are not widely spread in the community (for GFA2.0).

Many PVGs construction tools provide a graph in GFAv1.1 format, such as MiniGraph/Cactus [6], PGGB [5] or vg [7]. In addition, different tools offer functionalities for PVGs analysis, including basic statistics (gfastats [11], pancat [12]), basic manipulations (vg [7], odgi [13], gfatools [14]), subgraph extraction (pancat [12]), and retrieval of individual FASTA sequences (vg [7], odgi [13]). However, none of those tools provide an integrated suite for manipulating PVGs through a single, simple, and user-friendly

command-line interface. In addition, some tools exhibit limitations in terms of execution time or overall performance for downstream analyses, or require GFA files to be converted into their own format and this new format referred for all command, or to modify the initial file outside of the tool (e.g., node renaming) prior to analysis.

Here, we propose GraTools, an intuitive and fast command-line tool for manipulating PVGs using the initial GFA file as query name (see below), and conserving node names as input and output, in a transparent way for the user. It converts the graph in an optimized data structure once and allows for instance users to extract subgraphs and fasta sequences either from a given location (genome path or graph nodes) or based on a specific composition. It also provides statistics on the global PVG and on the Core/Dispensable compartment, as well as can perform basic analyses on the graph topology.

We provide here some examples as use-cases of increasing complexity in order to illustrate GraTools efficiency, using an Asian rice PVG [15] as example. While GraTools is primarily designed to efficiently work with acyclic PVGs coming from MiniGraph/Cactus [6], it also partially works with cyclic ones coming from other tools such as PGGB [5] after conversion (path to walk with coordinates). GraTools is developed in Python, installable using *pip* and an *Apptainer* container. The GraTools code and the supplementary scripts used here are available under the GNU GPLv3 license at https://forge.ird.fr/diade/gratools and https://data.ird.fr/dataverse/gratools, and the complete GraTools manual at https://gratools.readthedocs.io/.

2 Implementation

GraTools is implemented in Python 3.12+ and designed to be a robust and efficient tool for pangenomic variation graph analysis. It requires a single external dependency, BEDtools [16], which must be available in the system path.

2.1 Installation and manual

The tool can be installed *via* the Python Package Index (PyPI), from its source repository or using *Apptainer* and *Docker* containers. Documentation, including installation instructions, tutorials, and a detailed command reference, is hosted on *ReadTheDocs* (https://gratools.readthedocs.io).

2.2 One-Time Importing

All the commands of Gratools takes a GFA file as input. To enable efficient subsequent graph querying, each GFA is imported once in BAM and BED format (see below), either explicitly using the `gratools import` command, or automatically when any other command is invoked (if no `import` has been previously performed). During this step, the graph is parsed and stored in two optimized formats: an indexed BAM file for segments information, and a collection of BED files containing segments positions across individual genomes. The `import` phase is the most computationally intensive step, typically requiring from less than one minute to more than 30 minutes depending on the PVG size (see section 3.1). Once completed, subsequent analyses are performed

without re-parsing the graph, substantially reducing execution time. This internal file structure is fully transparent to the user, as said before, as the indicated input file for all the GraTools commands is always the initial GFA file.

2.3 Architecture and Core Components

GraTools is built on a modular, class-based architecture in which each component has a clearly defined role. A central main `Gratools` class orchestrates interactions with three other main classes, as illustrated in Figure 1.

Gratools Class

This class acts as the main orchestrator of all the GraTools commands, serving 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 their execution.

GFA Class

This class is responsible for the initial `import` of the GFA file, including its parsing and processing. It reads each line and extracts the relevant information to populate the internal data structures. During this phase, it forwards path information to the `AsyncBedWriter` for BED files creation, and sends link information to the `AsyncGfaDatabase` for database indexing (optional).

GratoolsBam Class

The `GratoolsBam` class manages all interactions with the BAM file generated during the initial GFA `import`. It handles essential operations such as writing, indexing and reading the BAM file, as well as querying it to extract specific nodes. This class forms the basis of the computational engine for most sophisticated commands. By leveraging the internal BAM file and the `pysam` library [17–20] in particular, it efficiently performs complex calculations such as core/dispensable ratios, segment depth statistics, or identification of shared segments between sample groups.

SubGraph Class

The `SubGraph` class extracts user-defined genomic regions from any embedded path in a graph. It intersects query coordinates in the pre-computed BED files using `pybedtools`[21], identifies the corresponding nodes and paths, and quickly reconstructs the subgraph in memory. The resulting subgraph can then be exported in GFA format (with native node identifiers), or as linear FASTA format.

AsyncGfaDatabase and AsyncBedWriter Classes

These two classes handles the various I/O operations during the `import` step. `AsyncBedWriter` creates BED files for individual genomic paths per sample, while `AsyncGfaDatabase` manages an asynchronous SQLite database to store and query all link data from the GFA file (optional). Batch insertion efficiently handles millions of links during the initial parse. Once the GFA file is read, internal files are created

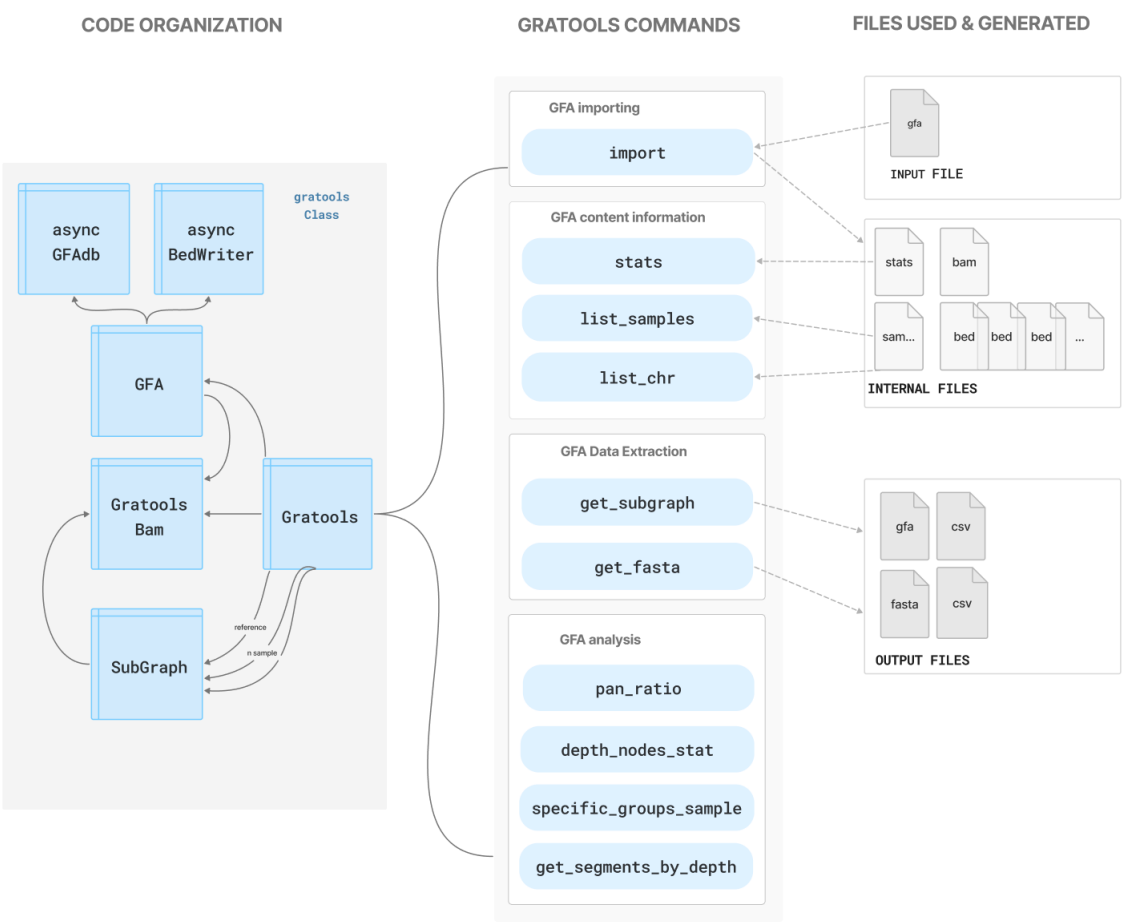

Fig. 1 Overview of GraTools: GraTools follows a class-based architecture in which the main `Gratools` class coordinating interactions with the three other classes: `GFA`, `GratoolsBam`, and `SubGraph`. The functionalities and commands are organized into four categories: **(i) GFA importing** (`import`), **(ii) GFA content information** (`stats`, `list_samples`, `list_chr`), **(iii) data extraction** (`get_subgraph`, `get_fasta`), and **(iiii) content analysis** (`pan_ratio`, `depth_nodes_stat`, `specific_groups_sample`, `get_segments_by_depth`).
The initial GFA file is used as input file, and outputs are provided in FASTA, GFA and CSV formats. Internal files (BAM and BED) are used by GraTools to speed up processing, transparently for the user.

to enable high-speed queries for graph traversal and connectivity, such as retrieving segments connected to a given node, or identifying distinct connected segments.

2.4 Performance and Scalability

Designed for large-scale pangenome graphs analyses, GraTools optimizes performance through asynchronous I/O, efficient data handling and parallel processing. I/O operations during GFA `import` are managed asynchronously through dedicated Python libraries such as `asyncio`, `uvloop` [22], `aiosqlite` [23], `aiofiles` [24], enabling simultaneous writing. After the GFA `import`, all PVG data are accessed through BAM and BED files, with BAM queries handled by `pysam` [17–20] and subsequent data manipulation on BAM or BED performed using `pybedtools`[21] or `pandas` [25, 26]. CPU-bound tasks, particularly those iterating over numerous samples, are parallelized using the Python library `concurrent.futures.ProcessPoolExecutor` [27], allowing the user to specify the number of cores and substantially reducing computational time for large analyses.

2.5 User Experience and Interactivity

A key design goal was to develop a command-line tool that is efficient, intuitive for most users, and transparent during its execution.

Command-Line Interface (CLI)

The CLI, built with the `Click`, `click-extra`, and `cloup` libraries [28–30], provides a clean, structured, and color-coded interface. Commands are grouped logically, with detailed help for each command and option, and shell completion facilitates rapid and accurate command entry.

Interactive Feedback

For long-running operations such as importing or parallel analysis, GraTools provides detailed, real-time visual information using the `rich` library [31] with multi-level progress bars that track both overall progress and individual subtasks (e.g., sample processing), giving clear visibility into execution steps progress.

Advanced Logging

GraTools implements a non-blocking logging system with log handling in a separate background thread, minimizing performance impact on the main application and supporting multiple verbosity levels with color-coded output.

3 Results

GraTools can perform a wide range of operations on PVGs from a GFA1.1 file (as described in Table 1) preferentially from MiniGraph/Cactus, many of which previously required many different tools (Table 2). However, in these other tools, most of the commands are multiples, requiring sometimes the use of another external tools, and referring to internally created files as inputs (Table 2). In addition, the node names may change between input and output. As described in Section 2.2, from the end-user perspective, in GraTools everything appears to be performed directly on the initial GFA file. The optimized internal data structures created by GraTools during

the `import` remain hidden, and outputs in GFA format preserve the original nodes, paths and haplotype names.

Based on the use-cases presented below, the GraTools features can be organized into three main sections: basic description of the graph, extraction of subgraphs and fasta sequences, and analyses of graph content and genomic diversity. The usage examples presented here are mostly based on the Asian Rice graph created in [15], which includes 13 accessions from [32], available at https://dataverse.ird.fr/dataverse/grannot and called **NewRiceGraph_MGC.gfa.gz**. We present 3 classical analyses, each illustrating a main type analysis, and a fourth example, requiring additional commands. These examples progress from simple to more complex use cases, addressing different biological questions.

3.1 Benchmark

All the computations were performed on a Intel ®Xeon ®Gold 6238R CPU @ 2.20GHz, with 56 cores hyperthreading and 512 GB of RAM, running under RockyLinux 9.1 BlueOnyx. We benchmarked different types of graphs (see Table 3) under different conditions of multithreading for the `import` command. As shown on Table 4, the use of 10 cores improved the speed of importing without increasing the RAM usage.

Compared to vg and odgi, GraTools is slightly slower when importing the Asian rice PVG (Table 5), with odgi being the fastest (8mn) and GraTools the slowest (24mn). In terms of RAM usage, GraTools `import` has the larger memory footprint (36 GB compared to 6 and 11 for vg and odgi, respectively). However, the GraTools `import`, once performed, can be used for calling regions from any of the embedded haplotypes in the graph, at the opposite of vg and odgi, where the index is strictly limited to a single "reference" genome for extraction (see below).

3.2 Description of the graph content

3.2.1 Obtaining basic statistics and importing

Similarly to many tools working with GFA files, GraTools proposes basic statistics about the PVG with the `stats` command including the number of nodes, their mean, minimal and maximal size, the mean and maximum number of output branching, and the total graph size. These statistics are computed internally during the `import` of the initial GFA; however, they can be re-called also directly through the `gratools stats` command, that will performed the `import` command if it has not been performed before.

Here, as a use case, we want to know which individuals are embedded in the Asian rice PVG, their chromosomes/contigs length and number, and the basic statistics on the graph. The command `gratools stats --gfa NewRiceGraph_MGC.gfa.gz` provides the basics statistics listed in Table 6. In a simple command, we have access to the mean segment size (32.44 in the example), the average degree (2.74), the total size of the graph (858 Mb), the distribution length of nodes, etc.

3.2.2 Listing the path names and the chromosome groups

The `list_samples` command will output the list of individual paths embedded in the graph, as shown in Table 7, while the `list_chr` will display the chromosomes (or contigs) list organized by sample (Table 8). These features can be helpful to quickly explore a graph, to know which samples (haplotypes) were embedded in it, or what are the homologous chromosomes/contigs that the graph builder considered (if any) to be grouped. The Table 8 provides also the fragments start and stop, *i.e.* the size of the genomic sequences for each part of the chromosomes/contigs embeded in.

3.3 Extracting parts of a graph

GraTools can use the coordinate system from any of the embedded sample (haplotype) in the PVG, without having to re-importing it, as it is the case in vg for instance. Indeed, the initial formatting of PVG under the GraTools data structure (`import` command) allows it to be able to recover any part of the graph based on any of these haplotypes, even if it has not been indicated as "reference" during the PVG building or importing. In the following section, we exemplified the subgraph and FASTA extraction using two coordinate systems, one based on the reference used for building the PVG (here the NipponBare genome called IRGSP in this graph), the other one from another haplotype (namely IR64).

3.3.1 Extracting a subgraph

While visualizing the whole PVG is quite unfeasible, having a view on a specific locus can help a lot understanding variations between haplotypes, using tools such as Bandage [33], as shown in Figure 2. The `get_subgraph` will, as its name indicates, extracts a subgraph based on the coordinates of *any* of the embedded path, not only based on the PVG reference coordinates. Here, using the Asian rice PVG from [15, 32], we extracted the Sub1 locus based on IR64 coordinates using the `get_subgraph` command, the PVG having being build using the Nipponbare genome as reference. This locus harbors three genes (Figure 2), the Sub1B (yellow part in Figure 2) and Sub1C (green) that are present on all samples, and the Sub1A (in pink) that exists only in 4 samples upon 13, including IR64, and is responsible of the tolerance to submergence in Asian rice [3]. We used thus the IR64 coordinates to extract the subgraph for this main locus of more than 150 kb.

The extracted subgraph can contain the whole set of paths embedded in the graph (`--all-samples` option), but it can be restricted to an user-defined list of them. The node names in the newly extracted subgraph GFA file are the same as in the original GFA file, for a better forward analysis. In addition to a GFA file, GraTools can also output the corresponding FASTA sequences for each (required) path by adding the `--build-fasta` option. The same subgraph can also be extracted from e.g. the NipponBare reference genome using the specific coordinates and the following command: `gratools get_subgraph -g NewRiceGraph_MGC.gfa.gz -sq IRGSP -chr Chr9 --start-query 6328847 --stop-query 6673971`. These two graphs are identical (available as SuppData).

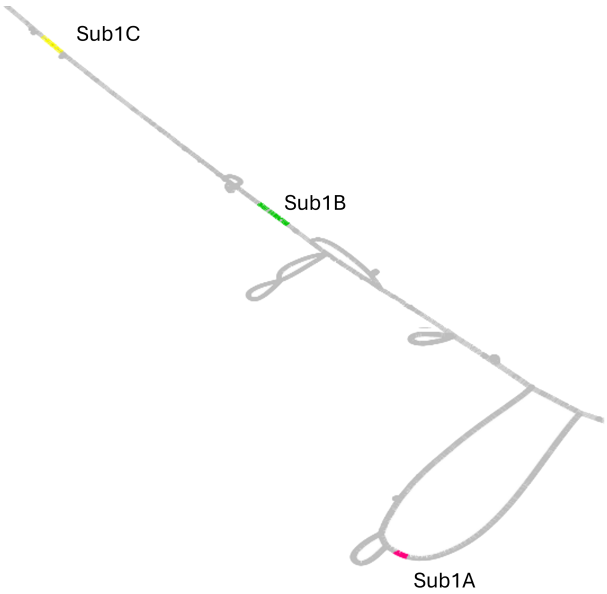

Fig. 2 Bandage layout representation of the Sub1 locus in the Asian rice PVG from [15] extracted based on the IR64 coordinates. The 3 genes of the locus, namely Sub1C, Sub 1B and Sub1A are depicted in yellow, green and pink, respectively on the graph structure. We used the following command: `gratools get_subgraph --gfa NewRiceGraph_MGC.gfa.gz -sq OsIR64RS1 -chr CM020884.1_OsIR64RS1_chromosome9 --all-samples --start-query 7400227 --stop-query 7800003`.

The time for extracting is partially dependent of the PVG depth (the number of embedded samples), as shown in Table 9. Indeed, extracting the same region in a larger graph is longer in CPU times (from less than one minutes to 5) with a memory footprint increase (from 0.16 to 1.3GB) when going from a 20 samples PVG to a 69 one. However, it is still completely acceptable in terms of time.

3.3.2 Recovering FASTA sequences from a subgraph/subregion

If only linear sequences for a given region are needed, the `get_fasta` command extracts the corresponding FASTA sequences for all embedded genomes based on the coordinates of one of the embedded genome. As with the previous command, the output can be restricted to a specified list of paths. The provided coordinates may also be defined from a path that will not be included in the final output.

3.4 Analyses on the graph

3.4.1 Core/Dispensable genome ratio

Most pangenomics analyses start with the estimation of the Core *vs.* Dispensable (or Pangenome) ratio. The `pan_ratio` command computes this ratio, while allowing users to define the percentage of haplotypes harboring a node classified as Core (Table 10). For example, a strict core can be specified with 100% sharing (no missing haplotypes),

while a more relaxed definition may require only 90% of haplotypes to share the node. In addition, the command can also report the proportion of specific sequences, defined as nodes present in only a small number of haplotypes. It also allow filtering of nodes according to their size (for instance, by computing the ratio only on structural variation larger than 50 bp, as shown in Table 10). In the current example here, for Asian rice, 61.81% of nodes larger than or equal to 50bp (`--fl` option) are classified as core (with a fixed core at 90%), whereas only about 30% of nodes are considered core when no size filtering is applied. Indeed, the threshold of the minimal size of 50 % removed 91.56% of nodes, indicating that the majority of nodes in the graph correspond to very small variant, typically SNPs.

Other commands linked to the depth of each node (meaning the number of haplotypes having this node in their path) exist in GraTools (Table 1), such as obtaining the statistics of node depths, or filtering out nodes with a minimal depth and size. All these commands can be restrained to a subset of embedded haplotypes, or excluding nodes under a specific threshold size.

3.4.2 Recovering segments based on group specificity

The `specific_groups_sample` command proposes to extract nodes that are specific from a set of paths compared to another, or shared by a set, or absent in it. Here, as before, the term "sample" refers then to a specific path from an haplotype in the PVG. 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 shows an application in deciphering the number of nodes that are core-specific from each of the two rice subspecies (*indica* and *japonica*) in the Asian rice graph used in [15]. Here, we launched a suite of 3 simple commands to obtain the strict core (100% of the paths having the node) from different compartment in the PVG, as follows:

- `gratools pan_ratio -g NewRiceGraph_MGC.gfa.gz --input-as-percentage -sm 100 -spm 10` provides the number of nodes shared by *all* samples (*Core shared*, Figure 3)
- `gratools specific_groups_sample -g NewRiceGraph_MGC.gfa.gz -sla samples_indica -slb samples_japonica` provides the *indica* core, *i.e.* the number of nodes present in *all indica* but *no japonica* (*Core Indica*, Figure 3)
- `gratools specific_groups_sample -g NewRiceGraph_MGC.gfa.gz -sla samples_japonica -slb samples_indica` provides *japonica* core, meaning the number of nodes present in *all japonica* but *no indica* (*Core Japonica*, Figure 3)

3.4.3 Advanced usage of the GraTools data structure

While GraTools propose a large panel of command-line analyses, it cannot cover all possible use cases. However, as its internal files are BAM and BED format, they can be directly reused for more advanced and specific analyses. Here, we present an example of such an analysis, which requires higher bioinformatics proficiencies than the previous analyses. We were interested in knowing the exact size of the pangenome, both in

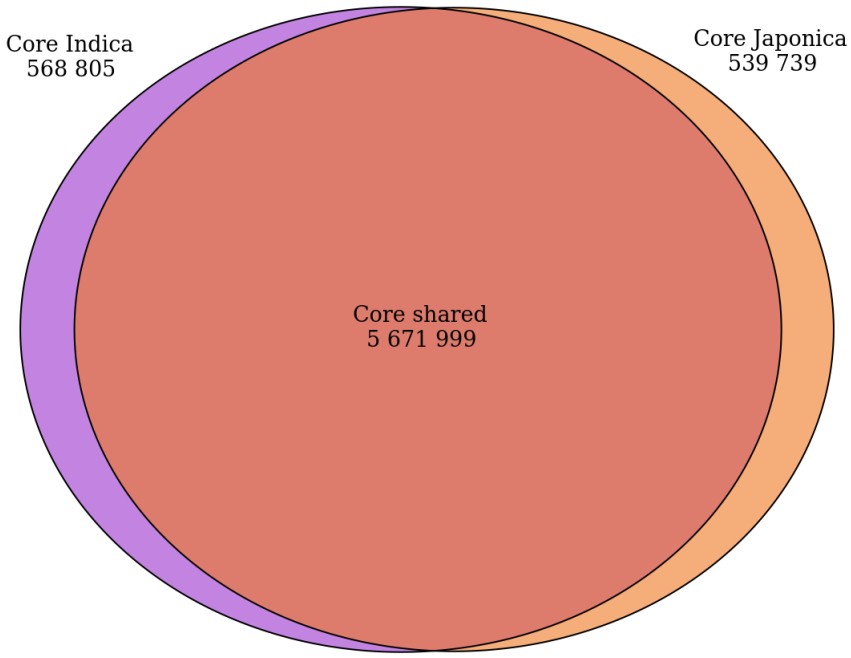

Fig. 3 Venn diagram representing the *indica* and *japonica* subspecies specific strict core nodes (shared by all samples) in the Asian Rice PVG from [15], computed using `specific_groups_sample` and `ratio` commands. For each subspecies (purple for the *indica*, and orange for the *japonica*), we indicate the total number of nodes shared by all samples. The intersection (Core shared) represent nodes shared by 100% of the haplotypes of the Asian rice PVG.

node number and total sequence length, for each of the two rice subspecies. All the codes used in this analysis are referenced in the "Availability of data" section 5.

We started by launching a series of `specific_groups_sample` command, one per sample, to identify nodes present in this individual but absent from all samples belonging to the opposite group. We then identified nodes present in at least one of the 8 *indica* but absent from all 5 *japonica*, and vice-versa. We then mixed the *indica* nodes to obtain an *indica*-only node list, and do the same for the *japonica*. Once these specific *indica* and *japonica* node lists obtained, we sum their size using a custom python script and the `pysam` API upon the BAM file containing the nodes information (see section 5 for the detailed commands). We then summarized the results as a Venn diagram, figured out in Figure 4.

As expected, the *indica* (8 samples upon 13) specific part is larger than the *japonica* (5 samples) one, as discussed in [32]. In Asian rice, the expected genome size if of 380 to 410Mb, and here the core part represent roughly 90-95% of the genome of each individual (371Mb). The mean node size in the core genome is 44bp, whereas it reaches

249 and 308 bp for *indica* and *japonica* part respectively. This indicate that most variations distinguishing the two groups correspond to larger structural variations rather than SNP, as seen in [34].

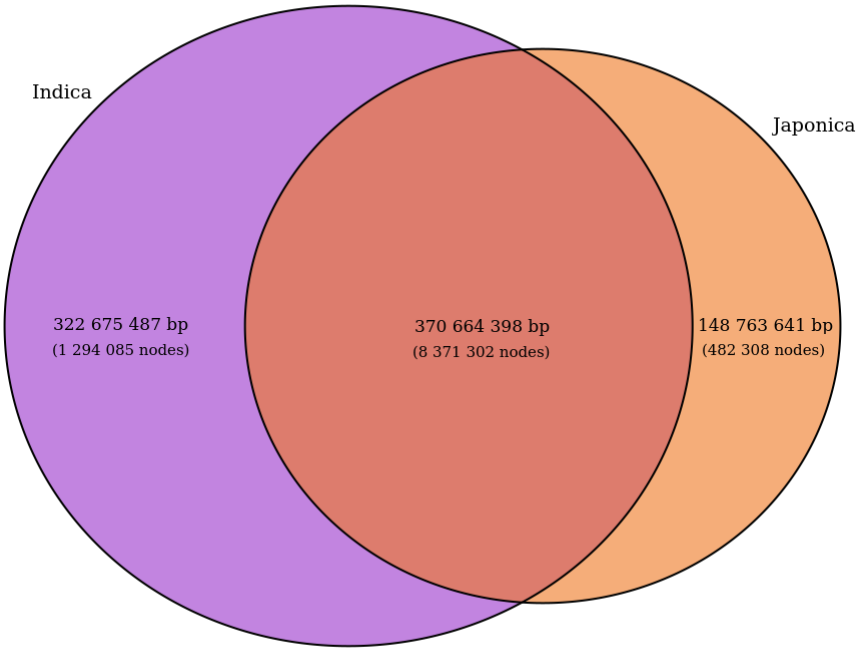

Fig. 4 Venn diagram representing the *indica* and *japonica* subspecies specific nodes in the Asian Rice graph from [15], computed using the `specific_groups_sample` command (see text for more details). For each subspecies (purple for the *indica*, and orange for the *japonica*), we indicate the sum in bp of the total nodes (specific or shared).

4 Discussion

Obtaining a PVG has become a classical step in the study of genomic variations across species ([8, 35]). However, once the PVG is constructed, structural variants (SV) are often extracted from the graph (mostly using the `vg deconstruct` command) to obtain a classical VCF file, capturing only a subset of the complex, nested SVs ([36]). Consequently, analyses frequently came back to linear VCF-based representation due to the lack of user-friendly tools that can directly exploit PVGs to answer different biological questions. To overcome these limitations, we developed GraTools to meet most of the needs of end-users with basic to advanced knowledge in pangenomics and

in command line usage. GraTools enables a rapid and innovative analysis directly on PVGs, including the computation of core and dispensable genome ratios, estimation of node depth across individuals, and extraction of sequences from genomic coordinates defined on any genome embedded in the graph. All these analyses are performed rapidly (e.g. `get_subgraph` command, see Table 9) once the `import` in GraTools format is done.

As GFA files are complex, huge tabular text-files difficult to manipulate natively, GraTools, similarly to vg [7] and ODGI [13], import the data contained in the GFA in dedicated optimized files. However, GraTools commands always refer to the GFA file itself as input, and not to these optimized files: it ensures a transparency and efficiency of usage for biologists. Moreover, the `import` command is intrinsic to any of the other commands: if it has not been yet executed and the BAM/BED files obtained when asking for another command (e.g. `gratools stats`), it will import the data in BAM and BED format automatically. In terms of performance for importing, GraTools is still indeed slower compared to vg and ODGI (see Table 4 and 3.1 part). This will be improved in the near future. However, GraTools is the only tools that does not require a new indexing when changing the system of coordinates, and thus is more efficient on the long term. Indeed, as shown in 3.3 and 3.3.2 sections, users can immediately change the path they use for providing coordinates, without having to re-import the whole data based on this "new reference". In addition, all these extractions refer to the same node name system, from the initial GFA file to all the extraction, ensuring a very efficient backward compatibility.

The GraTools commands (Table 1) are quite intuitive and streamed-line, compared to the multiple commands to perform for the other tools to obtain the same results (Table 2). They also permit to obtain rapidly information on the PVG and on the haplotypes embedded in (see 3.2.1 and 3.2 sections), extracting subgraphs (3.3.1 section), or to have a biological insight on their data (as for the Core/Dispensable analysis on Asian rice, 3.4.1 section). Finally, for power-users, the BAM and BED transformation of the GFA file give access to large results quite fastly (3.4.3). In this last case, we were able to obtain the data in less than one hour without any optimization of codes, a way much faster than the original analysis from [32], but with the same results.

5 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 eukaryotes.

Supplementary information. All Supplementary data are available at https://dataverse.ird.fr/dataverse/gratools.

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
- 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; CTD, FS and SR drafted the manuscript; all authors corrected and validated the manuscript.

Availability of data and materials

The rice pangenome graph used for figures and analyses is available in [15]

- Data availability: https://dataverse.ird.fr/dataverse/gratools
- Code availability: https://forge.ird.fr/diade/gratools for the most recent version, and https://dataverse.ird.fr/dataverse/gratools for the version used in the current manuscript

Availability and Requirements

- Project name: GraTools
- Project homepage: https://gratools.readthedocs.io/
- Operating system(s): Linux, MacOS, Windows (with WSL)
- Programming language: Python 3.12+
- Other requirements: Bedtools
- License: GNU GPLv3

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, Chichester, UK (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., 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

[6] 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

[7] 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

[8] 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)

[9] Consortium, T.C.P.-G.: Computational pan-genomics: status, promises and challenges. Briefings in Bioinformatics **19**(1), 118–135 (2016) https://doi.org/10.1093/bib/bbw089 https://academic.oup.com/bib/article-pdf/19/1/118/25406834/bbw089.pdf

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

[11] 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

[12] 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

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

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

[15] 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

[16] 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

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

[18] 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

[19] 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

[20] 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

[21] 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

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

[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] 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

[33] 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

[34] 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)

[35] Gaccione, L., Toppino, L., Bolger, M., Schmidt, M., Tassone, M.R., Sulli, M., Idahl, E., Alonso, D., Aprea, G., Ferrante, P., *et al.*: Graph-based pangenomes and pan-phenome provide a cornerstone for eggplant biology and breeding. Nature Communications **16**(1), 9919 (2025)

[36] Romain, S.: Identification, genotyping and representation of structural variants in pangenomes. Theses, Université de Rennes (November 2024). https://theses.hal.science/tel-05056938

[37] Dennu, L., Devic, M., Rigonato, J., Falciatore, A., Lozano, J.-C., Vergé, V., Mariac, C., Joli, N., Jaillon, O., Sabot, F., et al.: Biological and genomic resources for the cosmopolitan phytoplankton bathycoccus: Insights into genetic diversity and major structural variations. BioRxiv, 2023–10 (2023)

[38] Lian, Q., Huettel, B., Walkemeier, B., Mayjonade, B., Lopez-Roques, C., Gil, L., Roux, F., Schneeberger, K., Mercier, R.: A pan-genome of 69 arabidopsis thaliana accessions reveals a conserved genome structure throughout the global species range. Nature genetics **56**(5), 982–991 (2024)

[39] Kang, M., Wu, H., Liu, H., Liu, W., Zhu, M., Han, Y., Liu, W., Chen, C., Song, Y., Tan, L., *et al.*: The pan-genome and local adaptation of arabidopsis thaliana. Nature Communications **14**(1), 6259 (2023)

GraTools command	Description
GFA content information	
`gratools stats`	Compute diverse metrics characterizing graph size, connectivity, and complexity
`gratools list_samples`	List the unique name of all samples
`gratools list_chr`	List of unique chromosome names and counts per sample
GFA data extraction	
`gratools get_subgraph`	Extract a subgraph, based on a genomic region of any sample on the graph, for one or multiple selected samples.
`gratools get_fasta`	Extract sequences for a specified genomic region, for one or multiple selected samples
GFA analysis	
`gratools pan_ratio`	Compute the core-versus-dispensable genome ratio based on user-defined sharing thresholds, e.g., strict core (100% shared)
`gratools depth_nodes_stat`	Compute node depth statistics, reporting for each depth, the number of nodes, their ratio, and the count and the ratio of nodes passing optional length filters
`gratools specific_groups_sample`	Identify nodes that are shared within a specific group of samples or unique to that group compared to an optional second group.
`gratools get_segments_by_depth`	Retrieve nodes shared by a number of samples within a specified range, defined either as an absolute number or a percentage.

Table 1 Complete list of GraTools commands

	statistics	**content information**	**subgraph extraction**	**fasta extraction**	**segment extraction**	**import**
GraTools	gratools stat	gratools sample_list/ list_chr	gratools extract_sub _graph	gratools get_fasta	gratools get_segment _by_depth	gratools index
Gfatools	gfatools stat			gfatools view gfatools gfa2fa		
Pancat	Pancat stats		pancat isolate	pancat reconstruct		
ODGI	odgi build odgi stat*	odgi build odgi paths*	odgi build zgrep* odgi extract* odgi view*	odgi build odgi flatten*	odgi build odgi extract*	odgi build
VG	vg convert vg stats*	vg convert vg paths*	vg convert vg chunk/find* vg convert*	vg convert vg paths*		vg index/convert

Table 2 Comparative table of GraTools and other PVGs manipulation tools. Most GraTools feature are available on other tools, but the GraTools version is generally simpler and more explicit. For instance, the `odgi extract` command can select paths to extract, but it is node-based and not position-based, thus additional external commands (awk, grep,...) are needed to identify the desired nodes. In addition, post-import commands for odgi and vg (signaled with an asterisk) require the use the name of the newly created file as input and not the native GFA file name.

Graph	Pangenome size in bp	Nb of Samples	Nb of chr/sample	Reference
E. coli	11,630,600	13	1	[15]
B. prasinos	34,311,243	31	18-21	[37]
O. sativa	858,417,045	13	12 - 508	[15]

Table 3 Information about benchmarked PVGs in GFA A.1 format. They are all built using MiniGraph/Cactus, as in [15]. The number of samples and the number of chromosomes or contigs per samples are indicated.

Graph	Pangenome size	CPU, 1 thread	CPU, 10 threads	Output files size
E. coli	11 Mbp	00:00:43 (1.1 GB)	00:00:30 (1.2 GB)	381 MB
B. prasinos	34 Mbp	00:05:51 (8.3 GB)	00:04:50 (8.3 GB)	2.9 GB
O. sativa	858 Mbp	00:26:47 (36 GB)	00:24:31 (36 GB)	15 GB

Table 4 Benchmark of GraTools `import` command on various PVG listed in 3. We used different CPU number to test the multithreading efficiency. Results are in CPU time and in RAM usage (in GB, between brackets). The size of the resulting files in GB is provided.

Tools	vg convert	odgi build	gratools import
Time and Memory	00:08:40 (6 GB)	00:05:51 (11 GB)	00:24:31 (36 GB)

Table 5 Comparison of GraTools `import` vs odgi `build` and vg `convert`. All tests were performed using 10 threads and on the Asian rice PVG as described in Table 3. RAM usage is provided in GB, between brackets.

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	$1.04e^{-7}$
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

Table 6 Statistics on the PVG from 13 Asian rice genomes from [32] and built using MGC in [15]. The command `gratools stats` can be called directly from the native graph, or after an `gratools import` command.

- Total samples in GFA: 13
- Available Samples in GFA: NewRiceGraph_MG

Sample Name
AzucenaRS1
IRGSP
Os117425RS1
Os125619RS1
Os125827RS1
Os127518RS1
Os127564RS1
Os127652RS1
Os127742RS1
Os128077RS1
Os132278RS1
Os132424RS1
OsIR64RS1

Table 7 List of the embedded samples in the GFA from Asian rice, obtained through the `gratools list_samples --gfa NewRiceGraph_MGC.gfa.gz` command. General statistics are outputted before the table, in the terminal as in the CSV output.

- Total samples: 13
- Total unique chromosomes: 743
- Total fragments: 743
- Min fragment length: 28,006 bp
- Max fragment length: 44,959,450 bp
- Average fragment length: 6,878,547.43 bp

13 Samples, 12-508 Chromosomes/Sample (GFA: NewRiceGraph_MGC) - Full Fragment List

Sample Name	Chromosome Fragment Name	Fragment Start	Fragment End
AzucenaRS1	CM020633.1 _Azucena _chromosome1	0	44,011,168
	CM020634.1 _Azucena _chromosome2	0	36,468,344
	CM020635.1 _Azucena _chromosome3	0	38,425,022
	CM020636.1 _Azucena _chromosome4	0	34,192,892
	CM020637.1 _Azucena _chromosome5	0	30,954,872
	CM020638.1 _Azucena _chromosome6	0	31,189,045
	CM020639.1 _Azucena _chromosome7	0	29,327,599
	CM020640.1 _Azucena _chromosome8	0	29,168,739
	CM020641.1 _Azucena _chromosome9	0	23,125,300
	CM020642.1 _Azucena _chromosome10	0	25,095,745
	CM020643.1 _Azucena _chromosome11	0	30,181,525
	CM020644.1 _Azucena _chromosome12	0	27,487,302
IRGSP	Chr1	0	43,270,923
	Chr10	0	23,207,287
	Chr11	0	29,021,106
	Chr12	0	27,531,856
	Chr2	0	35,937,250
	Chr3	0	36,413,819
	Chr4	0	35,502,694
	Chr5	0	29,958,434
	Chr6	0	31,248,787
	Chr7	0	29,697,621
	Chr8	0	28,443,022
	Chr9	0	23,012,720
	ChrSy	0	592,136
	ChrUn	0	633,585
	chrC _11562 _osa1	0	134,525
	chrM _11706 _osa1	0	490,520
Os117425RS1	CM020694.1 _Os117425RS1 _chromosome1	0	43,678,955
	CM020695.1 _Os117425RS1 _chromosome2	0	35,923,432
Discontinued	...	...	...

Table 8 List (partial) of all chromosomes/contigs (called *Fragments*) from the embedded samples in the GFA from Asian rice, obtained through the `gratools list_chr --gfa NewRiceGraph_MGC.gfa.gz --full` command. the `--full` option provides the information of start and end of each Fragment. General statistics are outputted before the table, in the terminal as in the CSV output.

Graph size	20 samples	69 samples
CPU Time	00:00:45.46	00:05:47.37
(Memory footprint)	(0.16 GB)	(1.3 GB)

Table 9 GraTools `get_subgraph` command benchmarking. We tested o extract the same region (same coordinates on the Col0 reference genome), using a graph containing 20 samples and atother with 69 samplesfor *A. thaliana*. The exact command for the 69 samples graph was `gratools get_subgraph -g 69_arabidopsis_MGC.gfa -sq Col_0 -chr chr3 --all-samples -t 10 --start-query 8600000 --stop-query 8620000`. The PVG were built as explained in [15] based on the sequences from [38, 39].

- Total segments in GFA: 26,461,214
- Total segments analyzed: 26,461,214
- Total segments passing length filter ($\geq$ 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 $\geq$ 50bp)	1,380,679	2,233,791	61.81%
Specific (Dispensable) - Filtered (Length $\geq$ 50bp)	40,656	2,233,791	1.82%
Segments Filtered Out by Length	24,227,423	26,461,214	91.56%

Table 10 Core/Pangenome ratio statistics in the GFA from Asian rice, obtained through the `gratools pan_ratio -g NewRiceGraph_MGC.gfa.gz --input-as-percentage -sm 90 -spm 10 -fl 50` command. Here the Core is computed as present in 90% or more of haplotypes, the specific present in 10% or less of haplotypes, with an interrogation of fragments with a length larger or equal to 50bp. The *Raw* value are not filtered (*i.e.* all nodes)