Dear Dr. Hutchins,

Thank you for considering our manuscript for publication in Bioinformatics. We also thank the reviewers for their constructive criticism, and for the time spent reviewing this manuscript. We revised the text in response to the issues they raised. To ease the reviewing, we reported in red the changes in the manuscript. Furthermore, we report below the reviewer notes in black followed by the corresponding responses in red.

In summary, we made these revisions to the text:

- addition of keywords and conflicting interest statement
- extension of literature review to broaden community scope
- clarification of significance of our contribution, and limitation to pre-built graphs
- justification of ODGI data structures
- addition of a new tool for presence-absence variation mapping
- clarification of various GFA formats and ODGI's use of path coordinate systems
- sample naming standard used by ODGI and relevance for coordinates
- general extension of results to focus on vignettes and experiments
- description of subgraph extraction
- exposition of analysis vignettes related to human chr6, MHC, and C4 locus
- addition of C4A/C4B annotations to untangle alignment plots in Figure 2
- description of path Jaccard mapping (supplement)
- summary of path-guided SGD layout algorithm
- improvement of graph statistics results discussion
- more extensive performance evaluation vs. VG toolkit for common functions
- contrast between ODGI and other pangenome toolkits
- highlight of ongoing pangenome projects
- addition of supplement for detailed responses of general interest to readers

In general, we took the detailed comments from reviewers as an opportunity to greatly improve the manuscript. However, we did not make some suggested changes. For instance, we only made minor changes to Figure 1, which we note has actually been used by another community member in a blog post describing their lab's favorite papers of 2021 (https://www.treangenlab.com/post/2021-papers/#pangenomes). We have indicated other such cases in our responses.

A number of our responses are very detailed and may be of general interest, and perhaps should be appended to the supplement. We may ask for guidance from you and the reviewers in this aspect.

Regards,
Erik Garrison

Editor:

1) Conflict of Interest
As an integral part of the online submission process, Corresponding authors are required to confirm whether they or their co-authors have any conflicts of interest to declare, and to provide details of these. It is the Corresponding author's responsibility to ensure that all authors adhere to this policy.

None of the authors have competing interests. We added a section in the manuscript for clarification.

We added the ORCIDs of each author to the authors' list. This is provided in our attached TeX. Note that not all authors have updated their ORCIDs in the submission system, but we expect to resolve this should publication be recommended.

Reviewer: 1

Comments to the Author
Guarracino et al. present ODGI for the visualization, analysis, and editing of pangenome graphs. The intention is to make pangenome graphs accessible and useful for (biomedical) research. ODGI is available through github and a comprehensive documentation is available online. The topic of this work is of huge relevance, but I see several issues that should be addressed to make ODGI more useful to the community. Please find my specific comments below.

1) It seems that the keywords are missing. The authors might want to check this.

Thank you very much for spotting this. We added a keywords section in the abstract.

2) It would be helpful to clearly distinguish between the genome (DNA) and the genome sequence (information about the DNA). Most genome sequences are still incomplete representations of the genome and probably all existing genome sequences contain errors.

In the revised manuscript, we now distinguish between the genomes and the genomic sequences that represent them.

3) p2 l22-l33: This reads like another paragraph of the introduction. The authors might want to check if this was misplaced in the model section.

Thank you for pointing this out. The paragraph has been placed correctly in the introduction of the revised manuscript.

4) I am not convinced that the key property 1 (sparse graphs) holds for very large data sets which are the focus of this work. Especially when using pangenome graphs to represent entire clades (not just species), this could be violated. Gigabase-scale genomes should be particularly rich in repetitive regions due to genomic obesity (e.g. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC157029/).

Thank you. We are very happy to further elaborate on the sparseness of pangenome graphs:
"*A dense graph is a graph in which the number of edges is close to the maximum number of edges.*" (https://en.wikipedia.org/wiki/Dense_graph). Consequently, a sparse graph is a graph in which the number of edges is much less than the possible number of edges. As we allow self-edges and we have a bidirected graph, the number of maximum edges can be calculated with $2 * (\sum_{i=1}^{n} i - 1) + n$, where n is the number of nodes in the graph. One would classify a graph as sparse if the number of edges is at most half the number of maximum edges of that graph. The *HTTexon1* graph from Figure 3 of the paper has 35 nodes and 56 edges. So the maximum number of edges is 1225. The graph clearly is sparse. The centromere of a 90 haplotype human pangenome chromosome 8 (https://www.nature.com/articles/s41586-021-03420-7) graph has 377123 nodes and 560986 edges. The maximum number of edges is 142,221,757,129. The graph clearly is very sparse. Collaborators are currently building a *Cannabis sativa* pangenome graph from 12 haplotypes. Their current chromosome 7 graph has ~2M nodes and ~2.8M edges. The maximum number of edges would be 40,000,000,000. The graph clearly is very sparse. In the evaluation of the complex graphs above, we only observe very sparse graphs. Even the *C. sativa* graph, although each genome is to be expected to consist of ~75% of repetitive elements (https://www.nature.com/articles/s41438-020-0295-3), is very sparse.
Genomic obesity leads to more repetitive elements in a graph. In a variation graph, a repetitive sequence is usually compressed into one node with some edges connecting to the other nodes. The graph would get dense if a repetitive node would be connected to several million other nodes in the graph. We currently can't think of any biological data that would lead to such a phenomenon. Therefore, we argue that a large number of repetitive elements does not lead to a significantly higher number of edges, but to a significantly higher number of steps in the graph. And that's one of the major use cases ODGI was designed for: its *Step* structure and parallel path processing capabilities allow it to work with a very large number of paths and steps in a graph.
At some point in time, adding more and more genomes to an already very large pangenome will lead to (i) a core pangenome that will basically never change, and (ii) some individual genomic sequences that will add more sequence to the pangenome.

But, also taking the arguments of the paragraph above into account, adding new sequences even from complex regions like centromeres into a large pangenome graph won't lead to a dense, but a sparse graph, too. Ultimately, the construction method and the variation encoded in a pangenome graph have the greatest influence on the sparseness of a graph. Clearly, we can make a graph consisting of a very small number of nodes that represent, e.g., all extant 5-mers. This graph will not be sparse, but it will also be very different and serve a different research objective than pangenome, assembly, and multiple alignment graphs typically used in the research community.

5) There are already data sets with more than 1k different plant samples (e.g. https://doi.org/10.1016/j.cell.2016.05.063) or even thousands of human samples (https://doi.org/10.1073/pnas.1613365113). Solving the analysis challenge is not just relevant for biomedical research but also for plant and animal genomics. The authors might want to include a statement about the broad relevance of ODGI in the introduction or discussion.

Thank you for pointing this out. We have better specified the broad relevance of ODGI, also citing papers describing how pangenomic approaches can support and expand multiple genomic analyses in different species.

5) Is it correct that repeats are less likely to be accurately represented in ODGI? Since repetitive elements will be the focus of future genome projects, this would be very unfortunate.

This is not correct. The transformation of a graph in GFAv1 format to ODGI's binary format is lossless. Indeed, a graph in ODGI format fully represents all nodes, edges, paths, sequences, and any kind of variation present in the input GFAv1 file. Of note, the initial graph construction method itself, for example PGGB (https://github.com/pangenome/pggb) or minigraph (Li et al., 2020), determines the encoding of the repeats in the input graph. However, to avoid further confusion, in the revised manuscript we explicitly explain that. We also clarified this on our Welcome page of the ODGI Documentation (https://odgi.readthedocs.io/en/latest/).

6) Please start the y-axis of Fig.2 at 0 to clearly show that 16 threads are not reducing the required time to 0.

Thank you for this. As part of the manuscript's revision, we opted for a more comprehensive analysis of ODGI's performance. Thus, the original Figure 2 is no longer relevant and is now excluded. All y-axes in the new figures start at 0.

7) How does the run time scale with the number of haplotypes? Some additional benchmarking of the GFA conversion into the custom data storage format would be interesting.

In the revised version of the manuscript, we added a dedicated paragraph describing ODGI performance. In particular, Figure 4a shows the performance evaluation when converting the GFA file into ODGI's internal data format. Detailed performance measurements are available in the supplementary material.

8) How long does it take to visualize/draw figures for regions of different sizes and various numbers of haplotypes? It would be helpful to include an additional figure or supplementary figure that shows the run times.

We prepared a figure (Figure 4c) and a supplementary table (Table S3) reporting how the ODGI visualization capabilities scale with respect to the number of haplotypes.

9) How likely is it that users are actually able to edit a substantial proportion of a graph representing hundreds of genomes? Some semi-automatic feature could be helpful.

It depends on the complexity of the manipulation the users need to perform. For simple tasks, ODGI provides single commands for specific edit operations on the graph. For example:

- *odgi chop* divides nodes into smaller pieces preserving their order and graph topology.
- *odgi unchop* is the inverse operation of *odgi chop*, that is it merges nodes into single ones, preserving graph topology and node order.
- *odgi prune* removes complex parts of the graph, such as nodes that have a degree or a depth higher than a specified threshold.

For more complex operations, we already provide in the ODGI documentation a 'Tutorials' section (https://odgi.readthedocs.io/en/latest/rst/tutorials.html) where we guide users step-by-step to perform common analyses on pangenome graphs. Each tutorial shows how to use multiple ODGI commands in pipelines. This section is updated over time according to the pangenomic analyses we apply in our research. Among the tutorials available to date, 'Remove artifacts and complex regions' guides users to simplify graph structure to ease downstream analysis. For example, to perform short read mapping against the graph, users can follow this tutorial to remove collapsed regions in the graph generated by highly repetitive sequences in the samples.

10) In addition to GFA input, it would be helpful if reads could be imported from FASTQ/FASTA files to remove the need for individual de novo genome sequence assemblies.

ODGI can work with any kind of graph in the GFAv1 format, including graphs constructed with e.g. *vg construct* containing short read data. However, ODGI does not construct nor extend existing graphs. We clarified this in the Introduction and the Discussion of the paper. We also added a Warning on the Welcome page of the Documentation (https://odgi.readthedocs.io/en/latest/). ODGI can interrogate existing graphs, but does not support the importing of sequences in the FASTQ or FASTA format, which is out of scope. But, there already are other specialized tools to integrate such sequences:
1. graph constructed from long read or sequence data, extension with short reads or sequences: as the graph might be very complex, a necessary step might be to prune the graph with *vg prune*. Or removing complex regions following the ODGI tutorial for removing artefacts and complex regions. These steps might be necessary in order to build the indices required for Giraffe. Then map the sequences to the graph with *vg giraffe*. The resulting GAM file can be used with *vg augment* to extend the existing graph with the mapped sequences.
2. graph constructed from long read or sequence data, extension with long reads or sequences: here, our recommendation is to actually rebuild the graph with PGGB. One could use Graphaligner to align the long sequences against the graph and then use *vg augment* to extend the already existing graph, but that would be comparatively inexact. A reference-based method would be Minigraph followed by Cactus.
3. graph constructed from short read or sequence data, extension with short reads or sequences: map the sequences to the graph with *vg giraffe*. The resulting GAM file can be used with *vg augment* to extend the existing graph with the mapped sequences.
4. graph constructed from short read or sequence data (using a de Bruijn assembler for instance), extension with long reads or sequences: use Graphaligner to align the long sequences against the graph and then use *vg augment* to extend the already existing graph.

All of the above methods produce a pangenome graph in GFAv1 format which can then be analyzed with ODGI. We added the above information into our FAQs of the ODGI Documentation [https://odgi.readthedocs.io/en/latest/rst/faqs.html#how-can-i-import-reads-from-a-fastq-or-fasta-file-into-an-existing-graph].

11) How are heterozygous samples handled? Is it possible to process two or more associated haplophases? This would be especially relevant for polyploid species.

The GFA format can support metadata information. However, pangenome data models must work seamlessly across various legacy formats (e.g. FASTA, GFF3, BED, BEDPE, VCF, PAF, MAF, SAM) which do not natively allow for the association of sample identity with reference sequences. Every embedded sequence in a variation graph model can be used as a reference. To overcome this limit, we store biosample information in the sequence names that become the path names in the graph (and corresponding FASTA sequence records), by following the PanSN-spec convention

(https://github.com/pangenome/PanSN-spec). In more detail, we apply the following sequence naming scheme for sequences:

```
[sample_name][delimiter][haplotype_id][delimiter][contig_or_scaffold_name]
```

Where each field is optional. For instance, by using the character '#' as delimiter, the sequence name 'HG002#1#ctg1234' names 'ctg1234' on the first haplotype (or phase group) of the HG002 individual, while 'HG002#2#ctg9876' is contig 'ctg9876' on the other haplotype of the same individual. This can be naturally extended and applied for polyploid species as well. We added this explanation to the ODGI Documentation [https://odgi.readthedocs.io/en/latest/rst/faqs.html#how-is-heterozygosity-handled-by-odgi-how-polyploidy]. It is our recommendation to researchers working on pangenome collections to adjust their sequence names at the beginning of the analysis, to avoid confusion at diverse stages of analysis (alignment, graph construction, variant calling, MSA examination). Following this pragmatic and backward-compatible technique, ODGI will then allow for easy dissection of the pangenome based on sample information.
To clarify this approach to the community, we are in the process of proposing a formal extension to the GFAv1 spec that asserts a standard prefixing pattern to name samples and haplotypes. We anticipate that this will yield GFAv1.2. This PanSN approach has the advantage of harmonizing the GFA path names with sequences in FASTA format that is taken as input to graph construction in seqwish (Garrison et al., 2022) and pggb (https://github.com/pangenome/pggb), and allows a common data system that covers BED, VCF, GFF, and any other legacy reference based formats.

12) After reading the paper I am not sure who the targeted user should be. Basic computational knowledge is required for the installation. However, the inspection of individual regions seems to be a task for biologists/medical researchers. It could be helpful to explain how the intended users can actually use ODGI.

Almost all software in bioinformatics today requires basic computational knowledge for deployment and use and most sophisticated analysis requires scripting in languages like Python/R. For biology/biomedicine, this means that researchers who can use this software and also program in Python/R have a distinct advantage over their peers. Our ODGI tools analyze and visualize pangenomes and fortunately do not require scripting skills. Unfortunately, we don't escape deployment challenges, but with helping packager in Conda, for example, we limit the damage as it provides binary downloads.
As with most bioinformatics software, a bioinformatician usually runs the analysis and validates the results by comparing with existing methods or a biologist performs validation in the wet lab. The actual detailed biological interpretation of the results, for example, is usually left to the scientists with expertise in genetics, biology, or medicine. ODGI is exactly such a tool. Bioinformaticians can take a look at an existing graph, scanning for interesting regions. Then report back to the specialists with e.g. detailed plots who then can do a sophisticated interpretation of the analysis.

13) Is it possible to automatically extract all alleles of a given gene/genomic region? This could be a useful feature for researchers who are interested in a single gene across all individuals of a population.

Yes, this is possible. In the manuscript section 4.3, now Supplementary 7.3, in the first paragraph, we described how *odgi extract* is able to do this. In particular, *"regions of interest can be specified by graph nodes or path range(s), also in BED format. Furthermore, it is possible to indicate a list of paths to be preserved completely in the extracted graph."* Users just need to ensure that the list of supplied paths to be preserved in the extracted graph matches all individuals of a population. We performed this procedure e.g. for Figure 2a of the manuscript. In the ODGI documentation, we also provide a step-by-step guide on how to extract the MHC *locus* (located on human chromosome 6) while preserving all the haplotypes of the population represented in the pangenome graph: https://odgi.readthedocs.io/en/latest/rst/tutorials/extract_selected_loci.html#extract-the-mhc-locus.

14) Is it possible to quantify PAVs for a given gene of interest? This might provide an unbiased way to identify dispensable/core genes (https://doi.org/10.1186/s13007-021-00718-5).

Yes, it is. We have added a new command, *odgi pav*, which reads a PAV map given in BED format and produces the presence/absence matrix across paths or their groupings (e.g. samples or haplotypes). Here is a tutorial that explains the key functionalities: https://odgi.readthedocs.io/en/latest/rst/tutorials/presence_absence_variants.html. We thank the reviewer for this helpful suggestion. It seems to us to be a key functionality required by many researchers who work with pangenomes.

15) What are the memory requirements? I have not seen an assessment, but that would be very important to inform potential users about the required hardware. There are several pangenome projects that target thousands of samples (e.g. https://doi.org/10.1093/nar/gkw958). A table indicating the required memory depending on the genome size and number of samples could be helpful.

Thank you very much for raising this question. We agree with the reviewer. In the revised version of the manuscript, we added a dedicated paragraph describing ODGI's performance, including multiple tables in the supplementary section showing how pangenome size and the number of haplotypes affect memory requirements.

16) Similar information about the disk space requirements would be helpful.

In the supplementary section, we added a table (Table S5) reporting such information.

Minor comment:
p1 l52: "Consortium, 2018" ... is this formatted correctly?

Thanks for spotting this. The author of the citation is now formatted correctly. It now reads “Computational Pan-Genomics Consortium, 2018”.

Reviewer: 2

Comments to the Author
The manuscript from Guarracino et al presents their very nice tool ODGI, in the direct line of the VG toolkit. ODGI intends to help researchers to work with pangenome graphs, ie the most reliable representation of pangenomes so far. They introduced here numerous tools of various usage, from the simple stats to partial viz, including cleaning, QC and subgraph extraction. While the tool is efficient, of vary high interest and seems fast (I have tested it on the test data), my main concerns here is that the paper in itself is more a README/subtools manual than a scientific paper... Indeed the presentation is issued by subtools and not following a use case/user story, and thus I cannot figure exactly how to use it in different ways. In addition, I cannot find any benchmarks in terms of memory, CPU/disk usage or time for different types of data or in comparison with other tools such VG of gfatools. Finally, the paper does not clearly state already at the beginning that you need an already computed graph...

We apologize for the confusion. In the revised manuscript we now explicitly state in the Abstract that *“ODGI supports pre-built graphs in the Graphical Fragment Assembly format version 1“*.

Thus my main revision request (major) is to rewrite the manuscript with in mind the path a user has to follow to manipulate her/his own graph, with different examples and benchmarks. This will greatly improve the paper and the use of the tool.

We have structured the manuscript to explain ODGI's capabilities in a simple manner, with limited examples to ease their readability. In particular, we have reported the most frequent cases in which users find themselves having to deal. The choice is based on our experience and on the questions we are asked via the issues on GitHub, or via email and private messages. For example, one of the first tasks when having a pangenome graph is to visualize it for grasping its general properties, and we explained that in the 'Visualizing pangenome graphs' paragraph. If users need to study a specific locus, we

also explained in such a paragraph how we do that. And so on for the other paragraphs of the manuscript. Figure 2 in the revised manuscript submitted shows a real analysis we performed: we built a chromosome 6 pangenome graph from 90 human fully resolved haplotypes, we extracted the MHC locus, and visualized it, identifying the regions of high variability (shown as bubbles in the layout in Fig 2.a). Then, we extracted the C4 region, visualized it, obtaining a rough, but clear view of the copy number variation status of the locus in the input population (Fig 2.b, c, d, e). We then visualized the same region also with Bandage, by adding annotation in its layout automatically thanks to ODGI (Fig 2.f, g). Then, we gave an alternative, but more precise, visualization of the copy number status of the C4 region by plotting odgi untangle output with an R script (Fig 2.h). All the scripts for the analysis are available at the repository associated with the manuscript (https://github.com/pangenome/odgi-paper). However, we still strongly revised the manuscript to give more space to examples and, in particular, command benchmarks. We agree with that reviewer that this additional information improves the manuscript and best shows the functionality offered by ODGI.

In details, my comment for revisions are
1- I will start with my last main point: why being strictly limited to GFAv1 ? why not GFAv2 or rGFA ? I mean it can be completely explained, but please add a sentence on it.

Although the GFAv2 format is a superset of the GFAv1 format (http://gfa-spec.github.io/GFA-spec/GFA2.html#backward-compatibility-with-gfa-1), it was specifically designed for assembly graphs. However, the fields required to losslessly represent a variation graph are already specified in the more frequently used GFAv1 format. Regarding the rGFA format, it requires a genomic sequence to be chosen as the reference sequence upon which all other sequences are related to for their representation in the graph. rGFA implies a single hierarchical model built out of a chosen reference. While this is something that we believe could be generated, our experience suggests that it is of limited utility. Embedding many reference genomes in the graph provides the same key functionality, without placing limitations on the graph topology or its mode of generation. In GFAv1 there is no such limitation and this is fundamental to implement reference-free approaches. We decided to clearly state this in the revised manuscript. Furthermore, we also added a paragraph in the FAQs section of the Documentation (https://odgi.readthedocs.io/en/latest/rst/faqs.html#why-is-odgi-strictly-limited-to-gfav1-why-does-it-not-support-gfav2-or-rgfa).

2- In abstract, you use variation graph instead of pangenome graph. Why this difference ?

Thank you very much for spotting this incoherence. Generically, a pangenome model is a data structure that represents the genomic sequences of a population, a species, or a clade. This can take many forms, and in ODGI we focused on the graphical one. In

particular, ODGI supports pangenomes represented as variation graphs, which are sequence graphs that embed linear sequences as paths through the graph nodes. We apologize for mixing the terminology. This has been fixed in the revised manuscript, where we also added in the Introduction the differences between pangenome graphs and variations graphs. In particular, we now state that "*A class of methods to represent pangenomes involves the sequence graphs (Hein, 1989; Paten et al., 2017), where homologous regions between genomes are compressed into single representatives of all alleles present in the pangenome. In sequence graphs, node labels are genomic sequences with edges connecting those nodes. A bidirected sequence graph can represent both strands of DNA. On this model, variation graphs add the concept of paths representing linear DNA sequences as traversals through the nodes of the graph (Garrison et al., 2018). For example, a path can be a genome, haplotype, contig, or read.*"

3- in the introduction, you stated that "community still lacks a toolset specifically focused on graph manipulation and interrogation". What about VG toolkit, gfaTools or minigraph ? Ppangolin and other bacterial tools ?

Thank you very much for raising this question. We revised such a statement with "... *capable of operating on gigabase-scale pangenome graphs constructed from whole-genome assemblies*". In our work in the Human Pangenome Reference Consortium (HPRC), we needed to develop ODGI to handle pangenome graphs representing entire human genomes. Indeed, repetitive genomic sequences like centromeres lead to complex graph motifs that are not handled efficiently in VG, as also shown in Figure 4 in the revised version of the manuscript and the supplementary tables. As already explained in Comment 1, ODGI does not work with reference-biased rGFA files, therefore we don't compare with gfatools. PPanGGolin and other bacterial tools like PanX are very specialized tools for bacteria. The latter doesn't build a graphical representation of a pangenome. However, it already has a very developed eco-system, which allows a detailed analysis of bacterial genomes using an interactive GUI. ODGI does not come with such sophisticated tools, but it can serve as a starting point to explore viral and bacterial pangenomes on a very large scale. We added a section in the Discussion reflecting on PPanGGolin and PanX. We also mention PanTools, which provides a similar kind of platform for working with pangenome models, but differs from ODGI in its use of a specialized de Bruijn graph model. ODGI in contrast is meant to be a generic tool for working on a common class of genome graphs that are constructed by multiple methods.

4- In the intro as in the paper, the authors focused a lot on human pangenome only. What about the cattle, porcine, soybean ones ?

We reported human examples because ODGI was originally designed and implemented from the need to represent, manipulate and visualize complex genomic sequences (due

to segmental duplication, centromeres, satellites sequences) of 90 whole human haplotypes from the Human Pangenome Reference Consortium (HPRC). However, this does not exclude its application to other species too. We do not have direct experience with cattle and porcine, but our colleagues and collaborator successfully applied ODGI on soybean, *Arabidopsis thaliana*, *Mus musculus*, *Saccharomyces cerevisiae*, *Helicobacter pylori*, and we have tested ODGI's scalability also on *Zea mays* and multiple fish species.
When talking about ODGI's broad application case in the Discussion, we also added references to the bovine, rice, cucumber, and soybean pangenomes.

5- Still in introduction, again a reference to a variation graph data model... please explain more

The terminology has been fixed in the revised manuscript.

6- A figure explaining the mathematical terms and the different parts, in particular the difference between degrees and depth in your paper, would help clarifying further points.

We agree that the terms *node depth*, *path depth*, and *degree* can be confusing. In the manuscript, we used *path depth* and *node depth* as synonyms. We corrected this and replaced all occurrences of *path depth* with *node depth*. We added an explanation for node depth in the manuscript: "*... high depth of path coverage of some nodes, the so-called node depth, …*". Regarding the term *node degree*, as with all graphs, it is the number of edges connecting to a node. Else, we can't quite follow what is meant with "the different parts". Do you mean node, edge, step?

7- End of model part, why refocusing on ODGI if VGtoolkit exists ? What are the advantages of ODGI compared to VG (again a lack of benchmarking). In the same idea is ODGI compatible only with VG or with other graph tools ?

We believe that research software systems must be focused on specific clear objectives in order to succeed. The VG toolkit has become a complex mix of research-quality software, requiring sustained software development and maintenance efforts. It has succeeded as a testbed for a wide array of ideas and methods important to pangenome analyses. However, it is extremely poorly documented. Its lack of focus on basic graph manipulation and interrogation led us to redevelop and refine many of the presented ideas in ODGI. Also, there are technical limitations to consider. As noted above, a major limitation of the VG toolkit is its incapability to efficiently handle large graphs with complex motifs. These graphs are the results of including in the pangenome all genomic sequences of samples that represent near telomere-to-telomere assemblies of many genomes. This particular technical limitation led to ODGI's key data structure and our implementation of it. In the revised version of the manuscript, we added a dedicated

paragraph showing ODGI performance and its comparison with VG toolkit. ODGI is compatible with VG and any other toolkit supporting graphs in the GFAv1 format. We hope that many other algorithm authors choose to work in this same common data basis, but develop their methods independently to ensure greater diversity of choice for researchers focusing on pangenomes.

8- The three key properties are really linked to the data type issued from human... Thus, a highly divergent genome such as mosquito (with 1.5 SNP per 2 bases) would be much more fragmented, as well as highly complex and repeated plant genomes

We believe that the three properties hold true also for highly divergent species, and indeed we already have users that apply ODGI regularly on plant pangenomes (*Arabidopsis thaliana* and soybean). Regarding property 1, *"a dense graph is a graph in which the number of edges is close to the maximum number of edges."* (https://en.wikipedia.org/wiki/Dense_graph). Consequently, a sparse graph is a graph in which the number of edges is much less than the possible number of edges. Although divergent and/or highly repetitive genomes lead to more complex graphs, they will remain quite sparse. Regarding properties 2 and 3, they are true until the genomes represented are also loosely related to each other. Naturally, the more divergent the genomes, the more complex the graph will be.

Finally, we are confused about the reviewer's comment, in that the given SNP density would imply at least 75% nucleotide divergence. Perhaps is this rate found in a population of genomes? Such a graph should continue to be quite sparse, and although there are overheads due to the size of ODGI's node object, it does not violate any of our basic guiding assumptions. However, we do expect it would be difficult or nearly impossible to generate a meaningful graph where the pairwise nucleotide diversity is greater than 30-40%.

9- The closely related node in sort order thing is quite nebulous for me... If I follow your explaination, the sorting is performed by the ID of the node, ID defined as the position of the nodes in the initial graph... If the graph is updated, thus the node position/order will change, and thus the ID ? and the Step ? Any modified/removed/added node will modify the data structure of all subsequent nodes ?

The sort order of the graph is the order in which nodes are enumerated. We can assign new node ids to change the sort order. We find that, because they are typically very sparse, sorting pangenome graphs can help to reveal underlying structures and patterns of variation. This is key for visualization and interpretation.

We apologize for the confusing explanation of the graph sorting. Most subcommands in ODGI require and verify that the input graph's node identifiers (IDs) are optimized, that is

compacted from 1 to N where N is the number of nodes in the graph. If this assumption is violated, *odgi sort* provides functionality to optimize the graph. This means that the first node identifier (ID) starts at 1 and the last node ID is the number of nodes. All sorting operations update the graph in place with an efficient ID rewriting algorithm. The graph is then updated in place. First, the node identifiers are normalized (from 1 to number of nodes) including the adjustment of the edges. Second, path information, including both path metadata that points into the start and end steps of the path, plus each step of every path, is updated, too. We point out that changing the node order does not change our coordinate systems based on paths. These will now refer to a new node ordering.

When we sort a graph, we switch the node IDs of the nodes according to the result of the sorting algorithm. For example, if a random sort was applied, all existing node IDs would be replaced with new, random ones (the largest node ID would still correspond to the number of nodes on the graph). We would update the edge and path information as described in the paragraph above. The reordering of nodes has a great influence on how the pangenome looks like (Supplementary Fig. S2). We added this explanation also in the Supplementary section “Graph Sorting”.

Any modified node would require an update of its edge-reachable nodes. Also, the affected paths crossing these nodes will have to be updated. ODGI is able to alter these specific nodes and edges of a graph, it does not require updating or even rebuilding other parts of the graph the whole graph.

10- The bits required for data storage are in linear maneer, meaning that if the graph increases of X, the data structure will be increased of the same step or order ? Can you provide disk/ram usage benchmark ?

We added a whole performance evaluation section in the manuscript. See Figure 4 and Supplementary Tables 1-5 for detailed analyses across different numbers of haplotypes in the graph to get an idea of ODGI’s scalability.

11- The figure 1 is unreadable... please simplify it in a way or another.

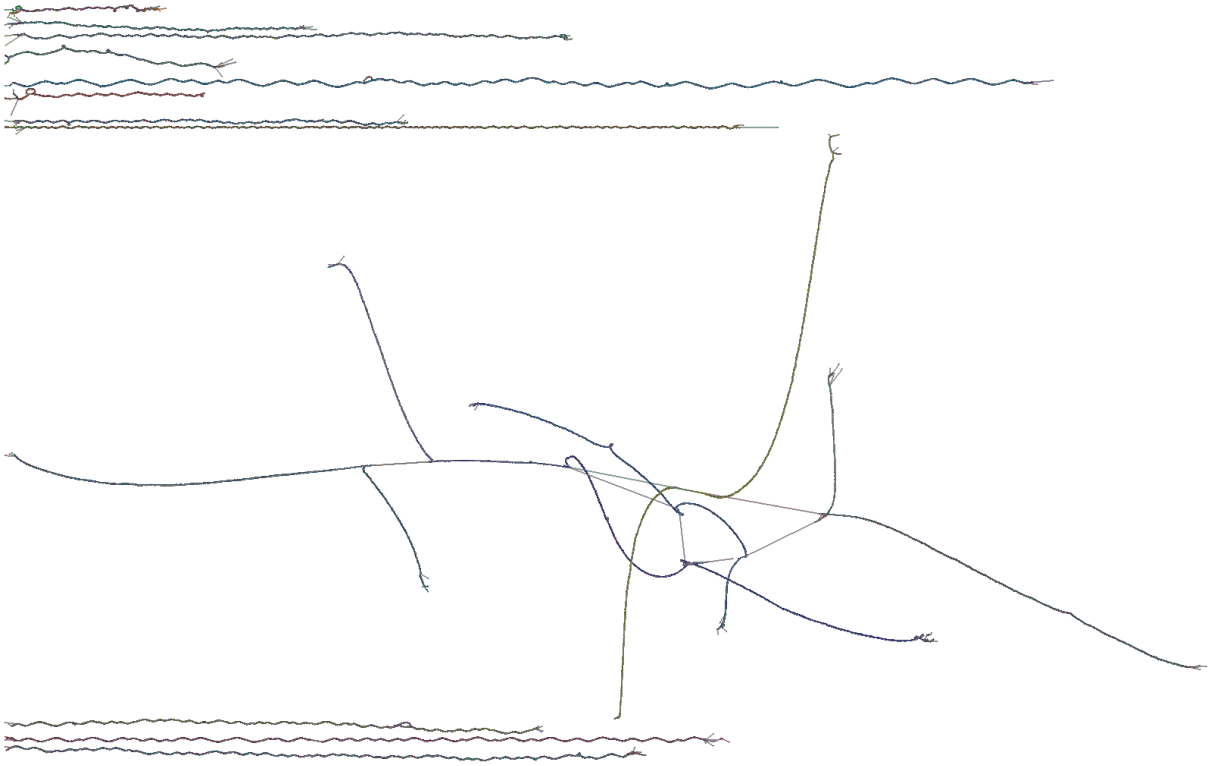
We slightly increased the size of Figure 1 and made sure to upload a high-resolution version of it. We would like to point out that our main intention is to share our manuscript in digital format, freely accessible online, therefore users will be able to zoom in on all the images. We would like to point out that although complex and compressed, near the end of our revision we discovered that this figure was chosen by a research member to represent the paper in a blog post summarizing their group’s favorite papers for 2021: https://www.treangenlab.com/post/2021-papers/. It shows the vast array of potential workflows enabled by ODGI, which is a key feature of this work.

12- The results part is a README part only... can we have any form of implementation for more than two or three tools ? with comparison with other tools ? in addition, the tools are listed in a given way here, then presented in another ?

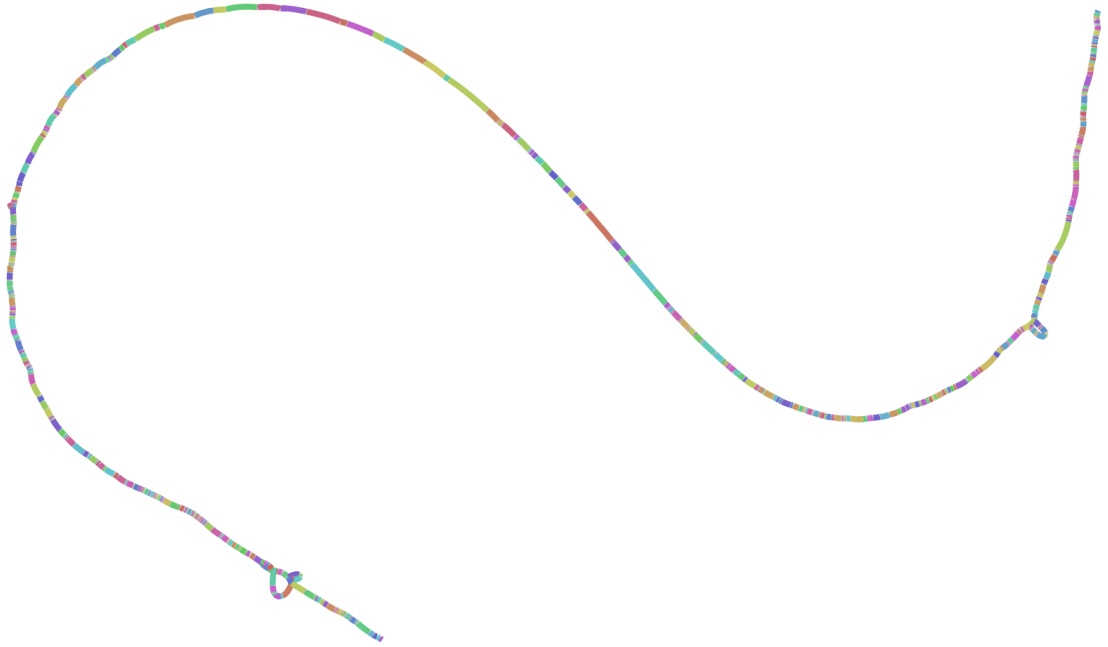
As already mentioned in the reply to reviewer number 2, comment major revision request, we revised the manuscript to give more space to examples and benchmarks focusing more on a “user story”. Moreover, we fixed the order of the tools and images in the exposition. We also moved text in the Results section that focused on explaining tools not shown in this section into the Supplementary section.

12- Why working only on chromosome 6 and not on the whole pangenome ? How will you thus include the translocations ?

Embargo requirements have limited our presentation of the HPRC data. As HPRC members, our data use authorization requires us to confirm with the consortium that any pre-publication use of the HPRC assemblies is acceptable. We were guided to present only a single chromosome. We believe that this is reasonable, and has sufficient scale and statistical power to be representative of significantly larger graphs.
Translocations are trivially represented in variation graphs. There will be a junction that the haplotypes containing the relative translocation follow, and this joins the two components of the graph corresponding to otherwise separate chromosomes. Although we do not present them in the manuscript, nothing in the data model we use nor ODGI itself will prevent the representation of translocations. They can easily be detected using *odgi untangle*, which projects the graph into a pairwise alignment format that can be parsed to find translocation events. As an example, here we report the 2D layout (obtained with odgi viz and visualize with odgi draw) of a pangenome graph made with 7 yeast assemblies (Yue et al., 2016):



The yeast genome is distributed in 16 chromosomes, but this pangenome graph consists of only 12 connected components. Indeed, the big component is composed of chromosomes VII, VIII, X, XI, and XII. If we extract the subgraph comprising, for example, a 10kbp-long portion of chromosome VII (between positions 10,000 and 20,000) of the S288 strain, we would obtain a subgraph containing paths from chromosome VII and XI. We visualize such a subgraph with Bandage in the following image:



13- Figure 4 is cited before figure 3

Thank you for spotting this. We fixed the numbering in the revised version of the manuscript.

14- The proposed viz are static ones, it must be stated.

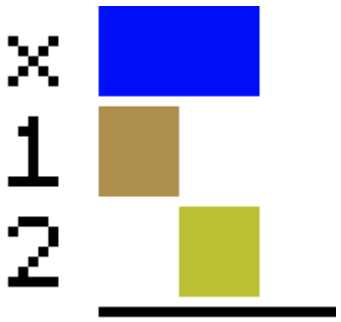
Thank you for this note. In the revised manuscript, we explicitly state that odgi viz and odgi draw "*provide scalable ways of generating raster images showing the high-level structure of large pangenome graphs.*"

15- "By default, path colors are derived from a hash of the path name" → What does that mean? How are they derived?

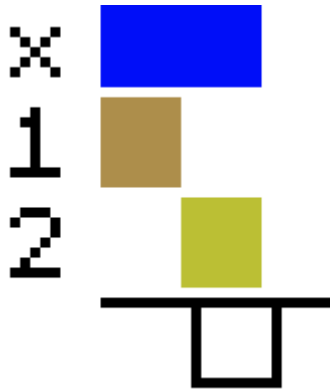
Path name strings are mapped to numeric values of 32 bits, from which we obtain the associated colors (each of the first three bytes corresponds to red, blue, or green). This simply ensures that when the same graph is rendered multiple times, the colors for the paths are stable. We have decided to simplify the sentence in the text for the sake of simplicity, as the previous level of detail was not fundamental for the odgi viz explanation. Now we simply state that "*By default, path colors are derived from the path names*".

16- "The black lines (...) represent the graph topology" → How ? it would deserve a better explanation, especially since it does not show all of the edges. The directly consecutive nodes are connected with links too, but they are not showed as black lines, so not all edges are represented by these lines. in addition, what about the unicity of nodes ?

We apologize for the missing information. In the revised manuscript, we expanded the odgi viz output explanation, better explaining how nodes and links are visualized. In particular, we state that '*Directly consecutive nodes are displayed with no white space between the two*'. Moreover, we explain in a dedicated paragraph in the Supplementary, that '*given a path X traversing two nodes A and B, the corresponding edge is represented by a black line starting from the left or right of node A if this node is traversed in reverse or forward, respectively, by path X; the black line ends to the left or right of node B if this node is traversed in forward or reverse, respectively, by path X. Consequently, if two consecutive nodes are linked both in forward, no edge is shown (it would be 0 pixels long, as it would start from the right of the first node and ends to the left of the second one).*'
Therefore, an edge connecting two directly consecutive nodes can't be displayed because it would be 0 pixels long. Here we report an example image made by *odgi viz* of the path (x) traversing two consecutive nodes (1 and 2) both in forward.



Instead, if the path traverses one of the two consecutive in reverse, for example the node 2, the line would be visible, as shown by the following image:



Regarding the unicity of the nodes, this is an aspect related to graph construction. Each node in the graph represents a genomic sequence found in at least one sample in the graph. For example, if all paths share the same genomic sequence, and the graph actually embeds such information (that is all the paths share such homology), this genomic sequence will be represented as a single node in the input GFAv1 graph, and it will remain represented as a single node in the ODGI format. In the *odgi viz* visualization, such a node will appear as a region in the images without any white rows, as white rows indicate where paths do not traverse, then share nodes.

17- Please define what the unicity of the nodes is here. Is a node with the same seq than another one folded? Does it appear multiple times in the ODGI format instead? It is important to understand the graphical representation.

Please refer to the previous answer where we discuss the unicity of the nodes. The graphical representation is of a variation graph (e.g. https://pubmed.ncbi.nlm.nih.gov/30125266/). We assume that such a graph is built from many complete assemblies. This is not a requirement for ODGI, but many of its tools benefit from the presence of path information across all nodes of the graph.

18- What are the limitations of this Fig 4 representation? How does it scale up ?

Regarding the representations produced by ODGI (Fig 4a, b, c, d, e -> now Fig 2a, b, c, d, e), the possible limitation is their level of detail, but it depends on the user's need. *odgi viz*, *odgi layout*, and *odgi draw*, the three commands used to generate them, are designed to work with gigabase pangenome graphs, without drawing base-level information of the genomic sequences embedded in the graph. If users need to visualize such information, that is the nucleotide sequences, they will have to use other tools, like *vg viz* (as we did in Fig 3.e), although such a tool doesn't scale to large graphs (indeed, Fig 3.e represents a small region). Similar limitation for the layouts produced by Bandage (Fig 4.f, g -> now Fig 2.f, g): such a tool does not scale to large graphs, and indeed we used it to represent only a portion of the MHC locus pangenome graph.

19-What are the “MHC and C4 pangenome graphs”? Say that these are human genome regions, at the very least. We are not all working on human…

You are absolutely right, we are truly sorry that we did not bring a short introduction on the subject. In the revised manuscript, we briefly introduced the Major Histocompatibility Complex (MHC) and the Complement Component 4 (c4) regions, also adding a specific reference.

20- still on figure 4 It is unclear whether Bandage is included or not within ODGI. Moreover, a ref is missing there.

We revised the description of Figure 4 (now Figure 2): now we state that the “*Layout of the C4 pangenome graph made with the Bandage tool (Wick et al., 2015) and…*”. Bandage's reference has been fixed too.

21- again fig4, panel H “Visualization” went to “Scatterplots”, and it is unclear whether they are done directly within ODGI. in addition, you jump to conclusion: “The plot shows the copy number status of the sequences in the C4 region with respect to the grch38 reference sequence, making clear, for example, that in HG00438#2, the C4 gene is missing”. Well, that’s unclear to me… First the plots do not show CNV, they show dots, corresponding to nt alignments between a genome and the ref grch38. The CNV can only be inferred from this. Moreover, there seems to be a deletion indeed in HG00438#2, but how does it make clear that it’s the C4A gene? Saying this assumes previous knowledge of what the coordinates of this C4A gene are. With just the graph, I would have no idea of what is missing exactly. So, can you explain to readers what it is, but the graphs do not, in any case, “make it clear”.

We apologize for the lack of annotations in Figure 4.h (now Figure 2.h). We totally agree that they are necessary for truly understanding the representation. Now, C4A / C4B gene positions in the GRCh38 reference have been highlighted.

22- Finally... how much time will it take to compute the figure ?

In the supplementary table S3, performance measurements are available when visualizing a pangenome graph of human chromosome 6. Since the MHC locus and C4 regions are a small subset of chromosome 6, the runtime/memory requirements for their visualization would be lower. *odgi layout* (16 threads) and *odgi draw* took 6:21 min in total to produce the 2D layout of the MHC locus on a 32GB RAM laptop.

23- In extraction regions, what about the interchromosomal translocations ?

The extraction process makes no biological assumptions. If the pangenome was built with multiple chromosomes together (that is, without partitioning the assemblies by chromosome), then the possible translocations would be present in the graph. In particular, translocations would create edges between the connected components in the graph. If a translocation between two chromosomes is present in the region to extract, then the extraction will involve regions from both chromosomes.

24- In the Editing part, it is said that groom will remove spurious inversions using the most prevalent paths. That means that in some cases the true minor variants will be removed ?

No, minor variants would not be removed. The grooming process is lossless with respect to graph content and overall topology: what is altered is the local orientation of the assemblies in the pangenome graph, with the aim of simplifying the graph structure for easier downstream analyses. Grooming works to simplify the representation of inversions, to require fewer edges that go between the two strands of the graph. We have added such additional explanations in the revised version of the manuscript.

25- In the part on the DAG discussion (page 4, up second column), it could be a good idea to discuss here how other tools deal with this (such as minigraph does...). As a main comment, there are very few citations outside of the group itself, may be a little cherry picked.

We took this opportunity to describe the pangenome coordinate system concept that we are using and compare and contrast it with that used in minigraph. In short, the approaches in ODGI use the coordinates in the entire set of genomes in the pangenome, while that in minigraph focuses on a single reference and coordinate hierarchy built on

top of it. Our approach places no conditions on the kinds or patterns of alignments used to build the pangenome graph.

In general, we added more citations of related methods, focusing on adding papers from other consortia, groups, and tools. We note that many existing methods for pangenome modeling are monolithic. For example, methods related to minigraph depend on particular properties of its graph construction algorithm. PanTools is similar, mixing graph construction properties (it uses a de Bruijn graph) and interrogation. ODGI is qualitatively different. It aims to provide only graph-interacting tools, focusing on the basic operations that are required to analyze and understand variation graph-based pangenomes. This should allow it to be usable in many contexts, provided methods that produce GFAv1-type graphs wherein paths represent embedded genomes or sequences.
We have chosen to focus on variation graphs due to their generality. As proof, note that in principle, a variation graph can embed the other mentioned models here. A minigraph can be projected into a variation graph model by adding the nodes of the minigraph (which define the reference-grounded coordinate hierarchy) as paths through an equivalent variation graph. A compacted de Bruijn graph can be transformed into a collection of contigs and alignments representing their k-1 bp overlaps, and this, in turn, can be transformed into a variation graph that contains the information in the original DBG, e.g. with gimbricate (https://github.com/ekg/gimbricate) and seqwish (Garrison et al., 2022). We are unsure if this particular point needs to be emphasized in the paper. We hope not, as we would like to maintain focus on the functionality provided by ODGI, and to fully make this particular point would require extensive digression in the work.

26- Can you explain more in details (or in supp data) the path jaccard index ?

We added an extensive supplementary section (Section 7.2.2) including a minimal example.

27- How are the projection of GFF done ?

That's an important question, thank you. A GFF file contains annotations for one or more paths in the graph. For each annotation, we know the start and end within that path. So we can annotate all nodes that are visited by such a path range with the information from the attribute field. If there are overlapping features, we append the annotation for each node. Using the same coloring scheme as in *odgi viz* we generate a color for each annotated node by its collected annotation.

If a subgraph was as a result of e.g. *odgi extract*, the path names are usually in the form of *name:start-end*. *odgi position* is able to automatically detect this and adjust the positions given in the GFF on the fly to the new positions given in the subgraph. For each GFF entry, it just subtracts the "missing" number of nucleotides from the start and end field. That's how we adjust for the subgraph annotation. We added a subsection in the

Supplementary. To also make this more clear for potential future users, we added a section in the FAQ of the ODGI Documentation (https://odgi.readthedocs.io/en/latest/rst/faqs.html#how-does-odgi-position-s-gff-liftover-work).

28- "ODGI provides a variety of sorting algorithms… the genetic variation they present." It is necessary here to provide examples or study cases showing that each sorting algorithm provides the chance to identify variations or structures and how each sorting algorithm works differently in this identification process.

Unfortunately, this would be out of the scope of the presented manuscript. Graph sorting is an open problem. The most promising algorithm we are aware of for the variation graph sorting problem is the path-guided stochastic gradient descent algorithm (PG-SGD), for which indeed we provided a dedicated brief explanation in the manuscript. This is an algorithm that we have developed and are still actively working on to improve its performance and output quality. We have planned to prepare a dedicated manuscript to explain more in-depth the graph sorting problem, how it affects pangenomic downstream analysis, and PG-SGD algorithm applications. Yet, we have added a figure in the supplementary (Figure S2) that shows more in detail the effect that the sorting algorithms have on the pangenome graph visualization.

29- Which are the metrics you refered to ? Can you procide supp data as examples ?

Unfortunately, we can't quite follow this question. Which section of the manuscript is it about? Which metrics are unclear?

30- Discussion: "It provides tools to easily transform, analyze, simplify, validate, and visualize pangenome graphs at large scale." can you justify it more ? The "large scale" is never discussed for visualization in this paper, while it is one of the main drawbacks I can imagine for the representation. How easily does it really transform the graph?

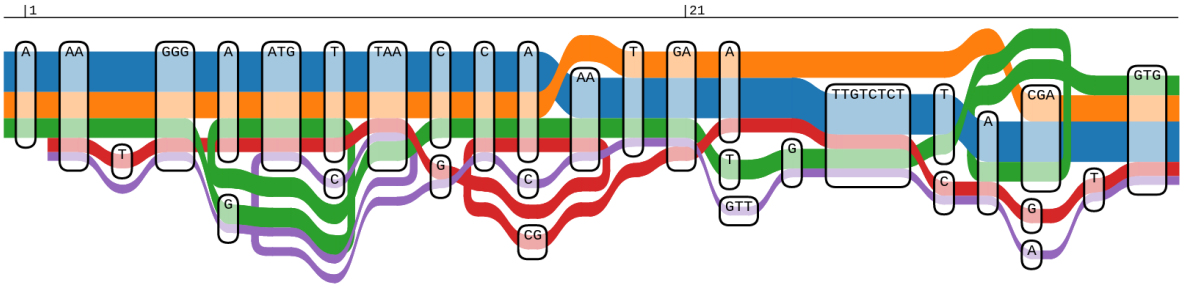
ODGI provides static pangenome graph visualizations and, as already replied in a previous comment, they do not report low-level details such as nucleotide sequences. This allows us to scale to large pangenome graphs. In the supplementary, we added a table (S3) reporting the performance measurements when visualizing a full human chromosome pangenome graph. We added text about graph transformations in the Supplementary sections 7.3 and 7.4 where we detail graph editing and graph sorting.

31- "Recent interactive browsers are reference-centric (...) or focus primarily on 2D" I do not agree with this statement. How come Sequence Tube Map is reference centric since

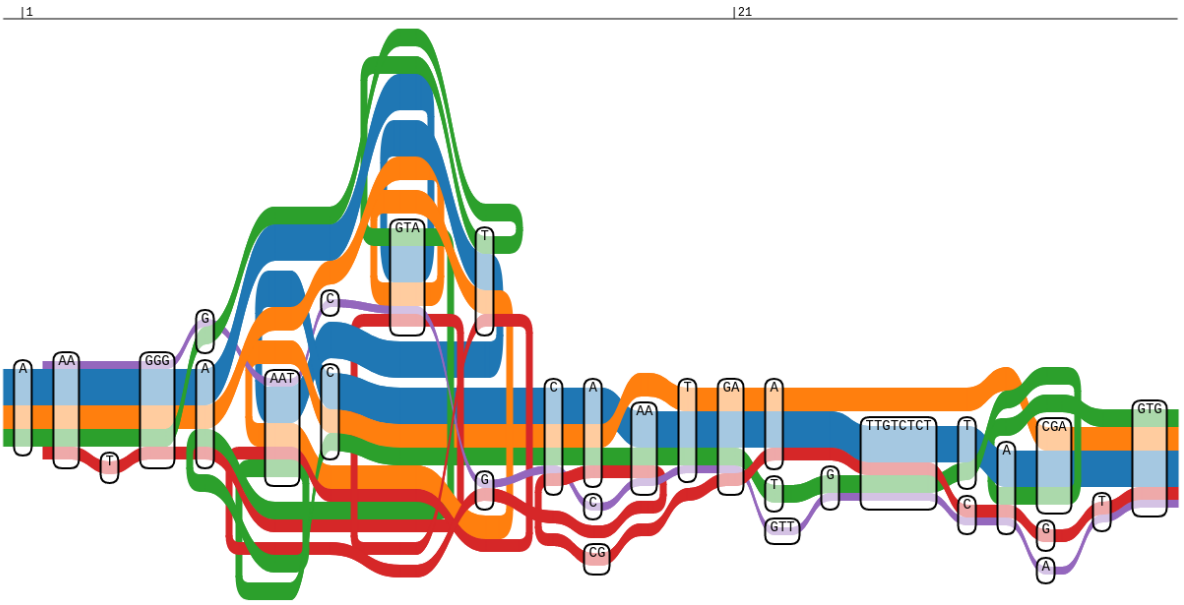
it shows graphs and paths alternatives? Panache is “order”-centric, just as the odgi viz representation is, but is not based on one reference genome. The paper from Liang and Lonardi is titled “Reference-agnostic representation and visualization of pan-genomes”, which directly contradicts the claim here, too. Please justify more your statement. Odgi vis enables changing the order of nodes easily, while the other tools might be based on pre-made files and that difference is interesting, but these tools cannot be called ‘reference-centric’ that lightly.

We apologize for this bold statement, which is wrong as it is in the submitted manuscript. Liang et al. 2021 is reference-agnostic, we should not have cited it here. However, the paper states that *“PGV depends on progressiveMauve for the multiple sequence alignment, which is computationally expensive. As a consequence, we expect about a dozen eukaryotic-sized genomes or about two dozen bacterial genomes to be a practical upper-limit to the analysis pipeline.”* So the method has computational limits, it couldn’t handle our data size. Furthermore, progressiveMauve does not create a graphical pangenome, nor it does support GFA format files (http://darlinglab.org/mauve/user-guide/files.html). Therefore, we excluded it completely from the manuscript.

Sequence Tube Map is reference-centric, where the visualization follows one selected “reference”. The input graph itself can be of a reference-centric or reference-agnostic origin. However, the actual layout is reference-centric. One path is chosen as “the reference” among all other paths which influences the layout. All nodes of the chosen path are drawn in a linear fashion, while all other variant nodes and paths are put where they fit. One can change this “reference” when double-clicking on a path of interest. This has a great influence when understanding a graph. For example, when running the most recent master with the example data, select “synthetic data examples” and “Inversions”. When the blue path is selected as the reference, it looks quite fine:



But when the purple graph is selected as the reference, the layout is totally different:



This demonstration is a major reason why Sequence Tube Maps is reference-centric.

While Panache's visualization style can be seen as reference-agnostic, the actual data it can work with does not have to be reference-agnostic. Personal communication with the authors revealed that, at the time of writing the manuscript, Panache was only applied to data in rGFA format. It also requires a BED file in order to build up its binary matrix, and a GFF3 file for the annotation. Subsequently, our and other scientists' trials to visualize GFAv1 data in Panache were very challenging (https://github.com/pangenome/pggb/issues/158). In fact, the current conclusion is that Panache is a tool that is limited to a predefined coordinate system (https://github.com/pangenome/pggb/issues/158#issuecomment-1046886677). This does not allow us to look at the graph from every sequence's perspective. Hence, we added "*... , have a limited predefined coordinate system, …*" to our statement.

In order to make it explicitly clear that we only want to generalize about pangenome graph visualization tools, we update the text in the Discussion to "*Recent interactive pangenome graph browsers are reference-centric, have a limited predefined coordinate system, …*". This ensures that we only meant pangenome visualizations that can actually work with pangenome graphs.