20/01/2024

PanToVA: automated preprocessing of pangenomes for variant analysis in PanVA

Bridging the gap between PanTools & PanVA

Vlugter, Sander
Supervision: dr. S. Smit (WUR), MSc. M. Yang (WUR)
Wageningen University & Research, Bioinformatics Group
Contact: sandervlugter@hotmail.com
Student number: 1160710

Abstract

A wealth of readily available genomic information set in motion a transition towards the pangenome approach which is set to replace the single reference genome, with the pangenome approach being able to provide a more complete picture of genetic variation. This higher resolution does come at the cost of increased complexity of the data structure of pangenomes, thus new tools like PanVA were needed to be able to visualise data of multiple genomes at once. However, with PanVA being a relatively new tool, proper integration into the workflow of comparative genomics using PanTools pangenomes was lacking. Resulting in non-standardised, unautomated manual curation and alteration of the pangenome before visualisation can commence in PanVA, adding yet another stage to the workflow, whilst creating more room for human error. To address this, we developed PanToVA, a tool that is able to handle any pangenome created with PanTools and preprocess it for visualisation in PanVA. In this paper we show PanToVA's ability to handle genomic data spanning bacteria, fungi and plants. In addition, PanToVA is integrated in PanUtils, a toolbox for automating the construction of pangenomes. With the introduction of PanToVA in PanUtils, the complete process from construction to visualisation of pangenomes is now brought together and automated. The implementation of PanToVA is made possible through collaboration with the developers of PanUtils, PanTools and PanVA.

1 Introduction

Ever since the first successful single complete genome was sequenced (Sanger et al., 1977), the technique of sequencing has been the basis of comparative genomics, gene discovery and function analytics using a single reference genome (Staden, 1982). However, the advances in (cloud) computing, data storage and next generation sequencing (NGS) of the mid and late 2000's made it cheaper and easier to produce significantly more sequencing data than was previously possible (Muir et al., 2016). As a result of these technological advancements large scale genome projects have been conducted, such as the 3.000 rice genomes project (Li et al., 2014). The increase of available sequencing data also highlighted the limitations of the standard single reference genome approach. Larger datasets revealed inter species variation in the form of presence and absence variations (PAV's), which cannot be explored using a single reference genome (Zhang et al., 2016). Additionally, the single reference genome is prone to cause reference bias in alignments, because of the tendency of sequences to map more easily to the reference alleles, whereas non reference alleles might be mapped at lower rates or be removed completely (Ballouz et al., 2019). Thus, continuing to individually compare each sequence in a dataset to a single reference genome is inefficient, costly and results in a significant loss of information.

To combat the issues of the single reference genome the concept of the pangenome was first proposed in (Tettelin et al., 2005). This concept describes the totality of all genes present in different strains. The term later evolved to mean the collection of DNA sequences present in a species, including a core part of the genome which is present in all individuals and a dispensable part of the genome present only in some (Marroni et al., 2014; Morgante et al., 2007; Sherman & Salzberg, 2020; Wang et al., 2023). This approach provides a more detailed and inclusive representation of variation in and even across species. It makes it possible to include sequence variants, phylogeny, phenotyping and (functional) annotations, which is valuable in comparative genomics within and across species. These pangenomes of multidimensional genomic information do come with a drawback. Analysing, interpretation and presentation of a pangenome is not as clear cut as with the single reference genome approach. Over the years various computational tools that are able to construct and analyse pangenomes have been introduced (Vernikos et al., 2015). However, efficient tools that can construct pangenomes alone are not enough to accommodate the complete transition to the pangenome approach (Hudson et al., 2010). In order to accomplish a complete transition, proper visualisation and presentation of pangenomes has to be achieved, as visualisation is a common method for interactive exploration and interpretation of

genomic data (Nusrat et al., 2019). In recent years several tools offering pangenome visualisation have been introduced. However, these tools do have their limitations, for example regarding scalability, accessibility, widespread applicability and in some cases the ironical reliance on the presence of the single reference genome which pangenomes are set to replace (Pedersen et al., 2017).

As it stands, to produce a sizeable pangenome with proper visualisation from raw data, users would need to use one tool for the construction and analysis and a separate tool for the visualisation. This is also the case for pangenome browser PanVA. PanVA (Brandt et al., 2022) is a promising new tool for visual exploration of pangenomes, designed to work with pangenomes created by PanTools (Sheikhizadeh et al., 2016), an established tool that is regularly updated and expanded. In the current stage, PanVA is able to visualise the various different data dimensions of pangenomes, handles decently sized (several dozens of genomes) pangenomes and is not reliant on the presence of a reference genome. However, PanVA requires users to manually curate the PanTools pangenome for entries PanVA can work with, as the initial version of PanVA was built on handpicked data. Users also need to make alterations and modifications to a preprocessing script that is necessary for pangenomes to be visualised in PanVA. Therefore, to make pangenome visualization by PanVA accessible to (unexperienced) users and applicable for pangenomes across the tree of life, we need to get rid of this manual preprocessing and automate this process in a flexible way.

To streamline and improve the connection between PanTools pangenome building and PanVA visualisation, we present PanToVA. PanToVA replaces the need for manual preprocessing and instead offers an efficient, configurable, automated and easy to use alternative able to handle any PanTools pangenome for visualisation in PanVA. In addition, PanToVA introduces extra features and filters allowing users more freedom to highlight specific data in the visualisation of pangenomes. In this paper we explain the design process and the choices made to deliver PanToVA. To demonstrate the wide applicability of PanToVA, we applied it to a collection of 8 PanTools pangenomes of bacteria, fungi and higher plants. These pangenomes vary in number of data entries, sequence lengths, phenotype and metadata. Finally, we discuss the advantages of PanToVA, the encountered limitations during the development, improvements made to PanVA and PanTools to be able to integrate PanToVA into the workflow, and planned future improvements.

2 Design and Development Approach

The PanToVA project consists of a design and development part as well as an engineering part. These parts come together to construct software that bridges the gap between PanTools and PanVA. As PanTools and PanVA were designed by different development teams, the data formats PanVA can handle are different from the way PanTools outputs the data. Therefore, to construct a proper connection some unorthodox design choices had to be made in PanToVA to be able to connect PanTools and PanVA. These choices could be viewed as ill-considered when examining the final workflow of PanToVA, without the context of the encountered issues. To be able to properly break down the reasoning for the design choices, this paper is structured following the design choices that led to the final version of PanToVA. Starting with section 2.1 which outlines the project requirements for the PanToVA workflow. Section 2.2 introduces the structural components and diversity of PanTools pangenomes, providing insights in the input data for PanToVA. Section 2.3 asserts the conditions for the data which PanVA accepts, and illustrates the expected end results of PanToVA.

2.1 PanToVA Requirements

To ensure the usefulness of PanToVA, it is essential to align its functionalities with the needs of the target user base. Therefore, several meetings were conducted throughout the duration of the project with both members of the in-house users and developers of PanTools and PanVA to identify and update these requirements. These sessions combined with the project proposal formed the basis for the project requirements.

The main requirement for PanToVA is to be **widely applicable**. Meaning it is essential for PanToVA to perform the preprocessing work for any pangenome built by PanTools of any organism. This also requires the input data prerequisites of PanVA to be addressed, because the sessions showed that only a limited number of pangenomes that PanTools built met the original criteria for the input data as set by PanVA. Pangenomes built by PanTools can contain a variety of additional useful data, which PanVA was not able to work with.

The variation in data present in pangenomes not only required modifications to PanVA but it also illustrates the need for PanToVA to be **configurable**. The original framework of the preprocessing left no room for customisation or varying pangenomes. So, PanToVA will allow users to customize, add, remove and highlight data entries to match their needs for visualisation in PanVA. As an added benefit, these customisation options greatly reduce run time and increase the efficiency of the preprocessing, as it will only run for the selected options and data points.

Improving **efficiency** is another requirement of PanToVA. As studies are increasingly adding more data to their pangenomes, and with the processing run time being correlated with pangenome size and available data, the swiftness of PanToVA is vital. However, speed must not come at the cost of efficiency. Thus, the design for PanToVA was made to balance run time and computational power usage. As a side objective for the efficiency improvements, the aim is for PanToVA to also be able to run properly on low-end desktops and laptops. In addition, improvements were set to be made to PanTools output formats and the PanVA code where needed, in order to improve the efficiency of the process as a whole.

Finally, the ultimate goal of PanToVA is to alleviate the burdensome task of manual data curation that is required for users of PanTools and PanVA. Therefore, paramount importance is placed on the **automation** of the preprocessing, **user-friendliness,** and accessibility of support. To achieve this, developmental versions were utilized in user test sessions throughout the project, which provided helpful insights to further improve the design of PanToVA. Additionally, based loosely on the MoSCoW method (Clegg & Barker, 1994) for prioritization, these valuable insights led to the inclusion of several additional features in the final version of PanToVA.

2.2 PanTools pangenomes

To get a better grasp of the complexity and diversity of data in PanTools pangenomes it is key to understand the data structure and the elements present. The data of pangenomes is often compressed and typically stored in a graph-based structure. PanTools builds a compressed De Bruijn graph, as conceptually shown in *Figure 1.A*, which is too complex for visual exploration of the data. When bringing down the resolution of the graph we can illustrate a more simplified version of the data structure of a PanTools pangenome in *Figure 1.B*. This view highlights the different datatypes and how all information is connected together.

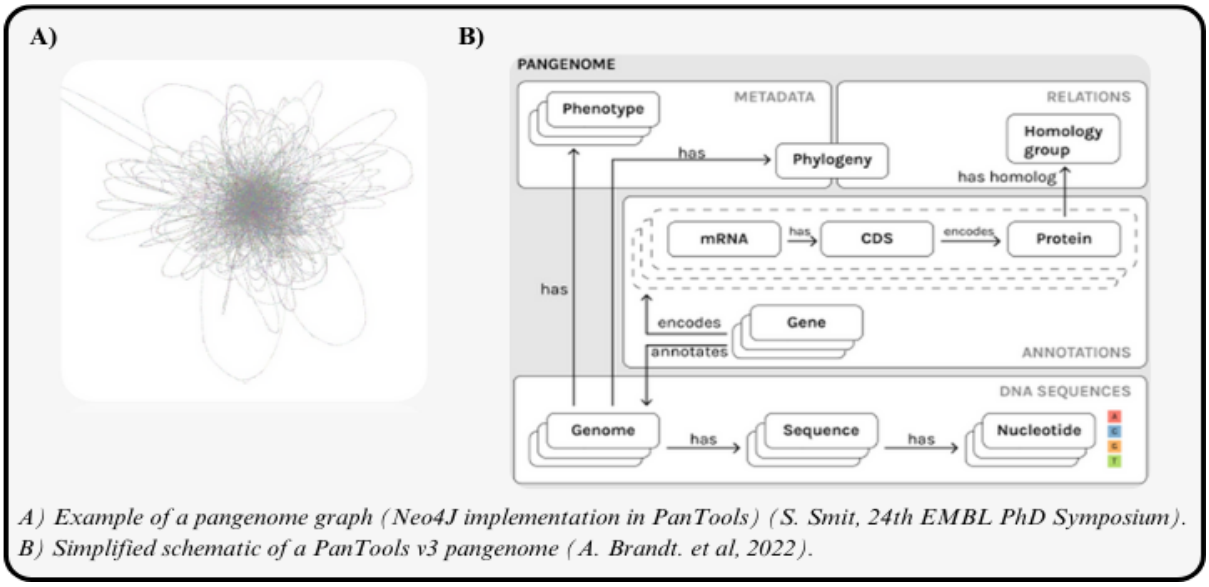


Figure 1. The PanTools v3 pangenome data structure.

Although the underlying data infrastructure of the pangenomes is identical, this does not mean the content of all pangenomes is the same.

These differences start with the design of and study for which pangenomes are constructed, as this heavily influences what data is present in a pangenome. For example, some pangenomes are constructed of a limited number of closely related species or strains in a specific habitat to gain insights in a specific population. Whereas others are built on a tremendous collection of genomes spanning multiple populations, habitats and or large evolutionary distance to gain insight in evolutionary relations. The organism(s) of interest affect the **annotations** in the pangenome, for example, eukaryotic and prokaryotic have differences and different standards in the annotations of genomes, which results in further differentiated data between pangenomes.

Continuing, the **type of sequences** of the genomes that are included in the construction also have an effect, as PanTools can construct a pangenome built of combinations of (multiple) reference genomes, genomes and resequenced accessions. However, PanTools handles data related to resequenced accessions differently compared to genomes. This causes more variance in the data structures of pangenomes.

The **metadata** part of pangenomes is optional to include with PanTools. Therefore, this information is not always available in pangenomes. Metadata also is a non standardised type of data, meaning that it can contain all sorts of added information. For example, it could include information such as coordinates of where the specimen was collected, date, time, the humidity and who took the sample, each as a separate category of the metadata. This also means that each category can have different datatypes, such as but not limited to numeric, scale, score, Boolean, degrees, coordinates, as well as descriptive text. In addition, PanTools accepts non-complete metadata entries, meaning it does not require every genome in the dataset to have metadata or even the same metadata categories to be included in the pangenome.

As part of the structure of a PanTools pangenome **phenotype data** can be added to the metadata block. Phenotype data, like metadata, adds extra information to the pangenome, however there are distinctions. Here we make use of the definitions as described by (National Human Genome Research Institute, 2020), which states that metadata are “data that provide additional information intended to make scientific data interpretable and reusable (e.g., date, independent sample and variable construction and description, methodology, data provenance, data transformations, any intermediate or descriptive

observational variables).". Whereas phenotype data are "data are the observable characteristics or traits of an organism or a cell line (i.e., the physical manifestation of a genotype).". Similar to the metadata, the phenotype data can be incomplete and are non standard.

Pangenomes are essentially a collection of genomes, and these genomes have relations that could span short and/or large evolutionary distances. PanTools can analyse and include information in pangenomes for the exploration of these relations through **phylogenetics**. To do this PanTools has 6 methods to create phylogenetic trees, each with settings that can be finetuned by the user. For any given pangenome none or several different phylogenetic trees can be included.

However diverse, all pangenomes are built on the concept of **homology groups**. To better understand homology groups we first focus on the individual genomes in the pangenome. Genomes contain genes which annotate the genome and encodes for mRNA. This mRNA has CDS (coding sequence) which encodes for proteins. All proteins found in the pangenome are compared based on their sequence similarity, forming groups of proteins called the homology groups. The individual protein sequences in a homology group remain connected to the genome to which they originate, which means any information that is connected to the genome is also connected. Hence, information such as metadata and phenotype data is available for the homology groups. As a result, traits such as phenotype specific attributes can be linked to specific proteins or perhaps to specific mutations. As these connections can be of great importance, PanTools has the option to include this information in the homology groups of the pangenomes.

Lastly, most PanTools functionalities that are executed when constructing a pangenome offer a selection of methodologies each with adjustable settings that can impact the data structures and connections. This once again illustrates the versatility and wide applicability of the pangenome approach. At the same time, this illustrates the diversity, complexity and interconnectivity of the data structures in pangenomes which complicates visualisation.

2.3 PanVA input

With the complexity, diversity and quantity of data offered by PanTools pangenomes, PanVA's design focuses on the visualisation of pangenomes based on the homology group level, allowing exploration of all genomes at once on gene level. The initial concept version of PanVA was developed based on the data, and more importantly the data structures, of 2 pangenomes. The first visualisations done by PanVA were completed using data that was manually retrieved from these pangenomes. To accomplish visualisation, the data from the PanTools pangenome were retrieved and transformed into several files to better fit the design of PanVA's visualisation approach. This as the output from PanTools was ill-suited for the web-based design of PanVA.

Additionally, with the first concept version some assumptions were made on data availability across pangenomes. This resulted in the implementation of features that were entirely dependent on the presence of data in a specific format, otherwise PanVA would not be able to visualise the dataset in its entirety. Throughout the PanToVA project, PanVA was updated several times to be able to account for these issues in order to handle the diversity present in PanTools pangenomes. This resulted in looser requirements as to what data is required and what data is perceived as optional. To accommodate these changes, the PanToVA code went through several iterations, partial rewrites, and additional preprocessing steps were required.

The final version of what is required and optional for visualisation of pangenomes in PanVA is illustrated in *Figure 2* and does not match with what PanTools produces. The preprocessing done by PanToVA results in these core files. The methodology of transforming the data structure and content of these files are explained in the individual sections of the PanToVA workflow. Detailed examples of these files are available on the PanVA github page (**Code & Data availability**).

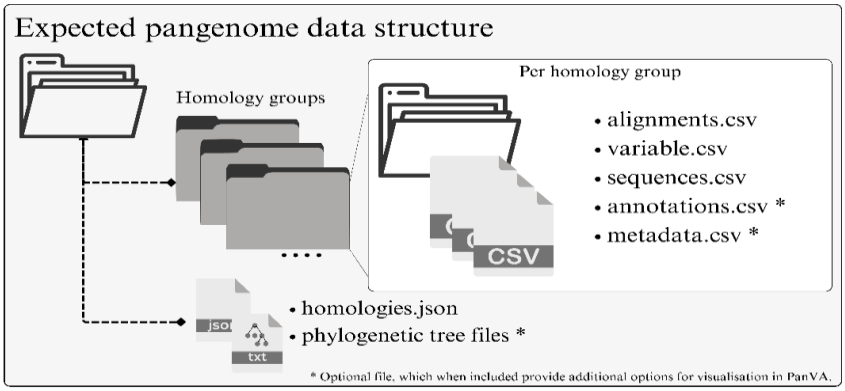


Figure 2. PanVA expected pangenome data structure for visualisation.

3.0 PanToVA stages of the workflow

Before preprocessing can begin, the pangenome needs to be constructed by PanTools. Changes were made throughout the project as to what PanVA requires and what is optional. This in turn affects which PanTools functionalities need to be executed when constructing a pangenome with the goal of visualisation, in addition to optional functionalities. Thus, aside from the standard functionalities used when creating a pangenome with PanTools, *Figure 3* highlights which PanTools functions are specifically required to be executed for visualisation in PanVA and which can optionally be executed to add more data to the visualisation in PanVA. In the paragraph following, we explains the choices made, encountered issues and inner workings of individual stages of the workflow, ending the chapter with the combined workflow.

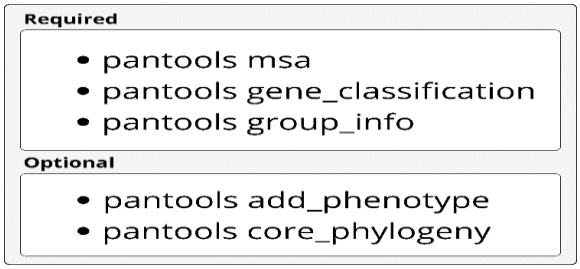


Figure 3. The Required and optional PanTools functions used to create a pangenome that can be processed by PanToVA for visualisation in PanVA.

3.1 Configuration file

After the pangenome is constructed the preprocessing begins the run by reading in the configuration file of PanToVA. This configuration file contains all the instructions PanToVA needs to start preprocessing a pangenome. Other methods of providing configurations to the program, such as parsing arguments on the command line and/or interactively were briefly considered. Yet the interactive method would be counterproductive given the focus on automation. Whereas providing all instructions in one command line argument would work, it would be prone to errors, because the number of (optional) arguments is sizeable which clutters the screen. The benefit of using a configuration file is the ability to retrace information regarding the run well after the preprocessing was initially done.

Within the configuration file, users are presented with several settings and options. Some settings are required, but do not affect the visualisation in PanVA itself. These settings consist of where the

pangenome is located, which group setting to use for the visualisation which corresponds to file locations in the pangenome and lastly the output directory. Others are optional and do not affect PanVA visualisation. These settings can improve the run time of PanToVA and provide additional information on the run when set to true. As one of the objectives is to improve the run time, PanToVA uses parallelisation allowing to spread the workload over multiple cpu cores when available. Therefore, a limit can be set for the maximum number of cores that can be used at the same time. This setting was added to improve run time for high end machines and allows low end machines to set a lower limit to prevent freezes and crashes. When the number of cores is not defined PanToVA checks the number of cores that are available at the start of the run and subtracts 2. If there are no free cores directly available, no maximum limit is set and PanToVA uses the cores when they become available.

Other settings do have an effect on the visualisation in PanVA when defined. These settings provide the user with some additional configuration options as well as data filters. The following sections will provide further details on these settings as these are optional and only come into effect in the specific stages of the workflow. Example configuration files are included with PanToVA as well as a reusable template version of the configuration file.

3.2 Data filters

The first module that is activated is the data filter. The data filter stage ultimately determines which homology groups will be visualised by PanVA. There are two filter types: one contains optional filters the users can adjust themselves, the other filters are required and cannot be modified by the users. The optional filters the user can interact with are set in the configuration file. Users are presented with a variety of data filters designed to give users the freedom to only visualise data of interest. One of the settings allows for the selection of only specific homology groups instead of all homology groups in the pangenome.

Subsequently the homology groups can be filtered based on minimal alignment length. The default setting for this is set to a length of 50 nucleotides. This threshold value is commonly used as the smallest measure when comparing (MSA) multiple sequence alignment tools (Nuin et al., 2006). Additionally, it was deemed unlikely that homology groups with an alignment length shorter than 50 nucleotides would be biologically or phylogenetically relevant, as PanTools makes use of MAFFT (Katoh & Standley, 2016) for the MSA, which does not consider the evolutionary relationships of the compared sequences (Ranwez & Chantret, 2020).

Users are also able to filter on the minimal number of members in homology groups. Here the number of members corresponds to the number of aligned sequences of the MSA assigned to a specific homology group. The default setting for this is 2, as it is impossible to have a multiple sequence alignment with less than 2 members. It is important to note that homology groups with less than 2 members do exist and are termed ‘single copy unique’, meaning that the sequence only occurs once in only one genome. These groups are interesting and informative, however PanVA is a tool for the visual variant analysis. If there are no variants found, they cannot be visualised in PanVA.

The other interactive filter option is the minimal number unique of members in homology groups. In contrast to the minimal number of members, here the number of members refers to individual genomes. This was incorporated, because homology groups can be found of which multiple or in some cases all sequences originate from one single genome. In other words: a genome might have a unique gene in relation to the pangenome with multiple (incomplete) copies of this gene, which makes MSA possible. These homology groups are classified ‘non-single copy unique’. The default setting for this filter is set to 1, meaning homology groups with sequences derived from one or more genomes pass the filter.

The filters the user cannot access or interact with have to do with checking the completeness of the data in the selected homology groups. This means that any homology groups which do not have the correct files they are removed from the selection. The number of groups removed in this manner is reported

back to the user in the log file at the end of the run and on the command line during the preprocessing. Additionally, homology groups are removed from the selection that do not have a trimmed multiple sequence alignment as this is a requirement set by PanVA. Lastly, any homology groups that do not have any variants in the alignment are also removed as this is another requirement determined by PanVA.

By filtering by the homology groups each sequence retains a connection with the genome they come from. This becomes important in the later stages of the workflow as PanTools provides metadata information in different ways depending on whether the sequences are derived from a reference genome, resequenced accessions or an unannotated genome.

Finally, the filter module constructs a log file of the run, which keeps track of which filter settings were used, how many groups were removed during which filter step and clocks the run time. This was done to improve reproducibility of the preprocessing, provide users a way to gauge the effects of the filter settings and to find, track and document any issues that may be encountered.

3.3 Pangenome metadata

The next module in the workflow is the metadata module, this module will only activate if the pangenome has metadata and the end user wants to include this in the visualisation. In a PanTools pangenome the metadata is stored in a single file called “phenotype_overview.txt”. In which for every genome their metadata and phenotype data is stored combined with an assigned genome number. However, if metadata is to be included PanVA expects this to be presented as a separate file “metadata.csv” for each homology group in the pangenome.

To create these files users must mark “pheno_info” as “TRUE” in the configuration file of PanToVA. This initiates PanToVA to retrieve the metadata and phenotype data from the pangenome file “phenotype_overview.txt”. The first hurdle encountered was due to the format of “phenotype_overview.txt”, as it was designed to be readable by humans. Therefore, the module has to parse the file in its entirety to retrieve the information of interest as this could be spread throughout the file. Second, PanVA expects individual “metadata.csv” files for each homology group. Thus, the information retrieved from “phenotype_overview.txt” parsing is stored in memory to be linked to the individual homology groups. Next, for all sequences in each homology group the fasta header is compared with the metadata information in memory, as this header contains the genome number linking the sequence and the genome. Following this, for each homology group a “metadata.csv” is constructed containing only the metadata and phenotype data of the genomes of which sequences are present in this homology group.

However, this is not the only method to add metadata information for visualisation. As mentioned, PanTools is able to add resequenced accessions to the pangenome, however by definition these are not complete genomes. Therefore, PanTools does not assign a genome number to any of these accessions, which makes it impossible to add any available metadata on these accessions in the same manner as for complete genomes by genome number. To accommodate users to incorporate metadata for these sequences if available, the configuration file option “reseq_meta” can be given a path to the metadata in .csv format as per the examples on the PanVA github page (**Code & Data availability**).

If either or both methods of providing metadata are used, the data of “phenotype_overview.txt” and the “reseq_meta” files are temporarily stored in memory as a single data entity “df_phenos” for use in the next stage of the workflow. This is done as it is computationally less straining to keep this relatively small amount of data in memory for the next stage compared to opening, reading and closing the individual files that were just created for each homology group.

3.4 MSA settings & metadata visualisation options

Continuing to the next module of "MSA/metadata visuals", not every pangenome will have metadata available for visualisation, which affects the preprocessing workflow. Moreover, the settings used when building the pangenome also affects the resolution in which metadata can be visualised. To manage these influences, three different modes of operation are set up in this module.

In mode of operation 1, the module activates if the metadata module has not been activated and thus no metadata is available. The module will then only create the files "alignments.csv", "variable.csv" & "sequences.csv" for each homology group as required in PanVA. With "variables.csv" indicating which positions in the alignment are variable and if these positions are considered informative. The indication of informative is derived from the pangenome and is depended on which thresholds are used in PanTools when building the pangenome. Any position that is variable, or variable and informative, are also flagged as such in the "alignments.csv" file.

The second mode of operation activates if the setting of "pheno_info" is set to "True" and "pheno_var" is set to "all". This mode is only available when the pangenome was constructed using the latest version of PanTools, using the function "msa" to include phenotype variants. This functionality applies phenotype and metadata information to individual nucleotides of the sequence on their position during the MSA. This nucleotide level of annotation consists of three categories defined by PanTools (Jonkheer et al., 2022); "phenotype exclusive", "phenotype specific" and "phenotype shared", as shown in *Figure 4*.

Homology group K-mer Function	Phenotype 1					Phenotype 2			Phenotype 3				Definition
	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	
1	1	3	1	1	2	1	1	1	1	1	1	1	Core
2	1	0	1	1	1	1	1	0	0	0	1	1	Accessory
3	0	0	1	0	0	0	0	0	0	0	0	0	Unique
4	1	0	1	0	1	0	0	0	0	0	0	0	Phenotype exclusive
5	1	1	1	2	1	0	0	0	0	0	0	0	Phenotype specific
7	1	1	1	1	2	0	1	2	2	1	0	0	Phenotype shared

Figure 4. PanTools phenotype specificity chart (Jonkheer et al., 2022).

For instance, a homology group alignment may consist of 35 sequences each from1 a unique genome. For these genomes information is available on which population they belong to. 12 genomes are from population A, 11 from population B and 12 from population C. If we then find that on position 100 in the alignment 10 of the sequences have a Cystine, and all these sequences are part of population A, this is considered to be a 'phenotype exclusive' element. Meaning Cystine is only found on this position for population A but not for every sequence from population A. Whereas, using the same distribution as before, had it been that 12 sequences were to have a Cystine on position 100 and these were all part of population A, it would be considered a 'phenotype specific' element. This leaves the 'phenotype shared' elements, in which cystine appears on this position for all members of population A, but can also appear on this position for all other populations.

This data is retrieved from the pangenome by PanToVA and applied to "alignments.csv" and "variables.csv" files required by PanVA. This is done, because information is added on nucleotide position level and PanVA requires this information to be included in the "alignments.csv" file and not as a separate file as provided by PanTools. Modifications were needed as the traditional method of storing MSA data PanTools provides cannot accommodate this, as can be seen in *Figure 5*. Before the

introduction of this feature, PanVA already expected the sequences in the MSA to be split on position, which ends up allowing the addition of the phenotype category labels per position.

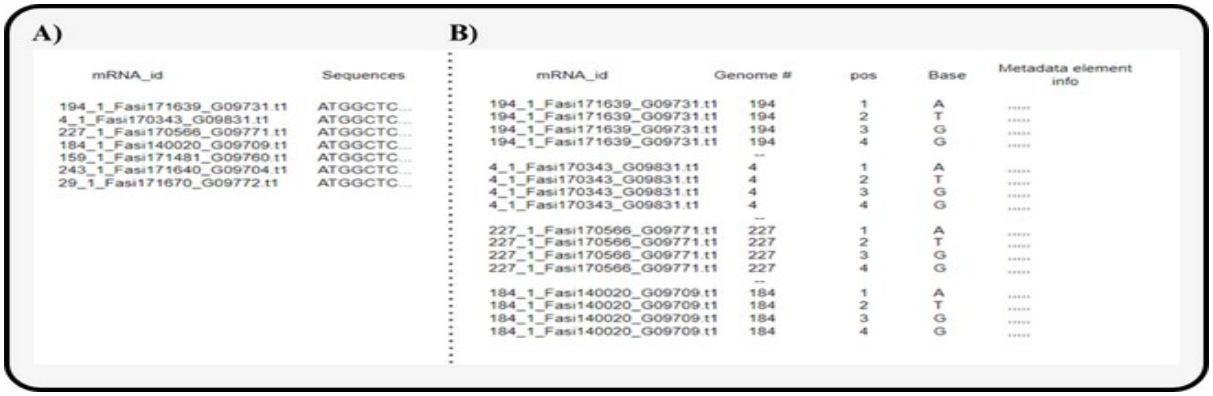


Figure 5. MSA file structure differences. A) Common approach for storing MSA data. B) PanVA approach to storing MSA data, allowing for position based information to be stored alongside the position.

Lastly, the third operating mode activates if “pheno_info” is “True” and “pheno_var” is set to “False”. This mode functions in a similar way as the second operational mode except it does not add information on individual positions of the MSA in homology groups. This was done because the feature to add this level of information to nucleotide positions is part of the latest update of PanTools and it remains optional. In addition, several smaller adjustments to the data formats were made to file formats that PanTools in between major updates. This resulted in several versions of files being constructed depending on which version of PanTools was used before the last major update. Where PanToVA aims to be widely applicable, user-friendly, and (re)building large pangenomes costs a considerable amount of time, it is beneficial if older pangenomes can also be preprocessed for visualisation. Therefore, this mode of operation provides users the option to only use metadata on genome level, makes it possible to use pangenomes built using older versions of PanTools and resolves all inconsistencies in the file formats of between versions.

3.5 Homology group characterisation

PanTools offers users the ability to add a variety of annotations on the sequences to the pangenome, from different sources such as GO, InterPro, PFAM, TIGRFAM. This information in combination with other metrics such as alignment length, number of variable positions and metadata flags is stored per homology group. Additionally, the classification of the homology group is extracted and stored with the different classifiers matching the PanTools classifications of; “core & single copy orthologs”, “core”, “accessory”, “unique” (the PanTools team, 2023). It is important to note that any homology groups with the label “unique” that make it through PanToVA, are unique to one genome but have at least two partial copies of a sequence. This is because single sequence unique homology groups cannot be aligned to another sequence as no variant exists in the data and thus these homology groups are not visualised in PanVA. The classification and metadata information is combined to form in “homologies.json”. PanVA uses this file to construct the control panel, and expects this to be formatted as a single .JSON file. From this control panel users can explore the pangenome by homology group on the metrics available in the metadata and by gene name, function and or classification.

3.6 Gene Variants & Annotated Genomes

During construction of the pangenome in PanTools the “msa” function is used. This function by default will only use nucleotide sequences of coding sequence (CDS) when used for pangenomes. When development of PanToVA started on this module the only method to alter this behaviour was by using

the PanTools option “add_variants” which, next to allowing users to add gene variant sequences to the pangenome, also created files containing information on gene annotation. This information consisted of intron and exon locations, and for exons what sections were consisting of untranslated regions (UTR’s) and coding sequence (CDS) which is particularly useful information for pangenomes of eukaryotic organisms, as prokaryotic organisms do not have introns. Initially, PanToVA would make use of the “add_variants” behaviour by reformatting the files containing the annotation information in a way that PanVA can visualise this information. The “msa” function has since been updated, to allow users to alter the “msa” settings to perform the alignment on the entire mRNA sequences. This creates the same data on the gene annotations as before thus allowing PanToVA to now also include this data without having to use the “add_variants” functions. These annotations however are still only available for annotated genomes, or when the “add_variants” option is used with variants from VCF (variant call format) files in combination with a reference genome, making this module optional in the preprocessing. Users can indicate if the add variants option was used or not by changing “msa_type” to “msa_per_group_var” in the configuration file of PanToVA. If left unchanged PanToVA will add annotations from annotated genomes in the pangenome if available.

3.7 Phylogenetic trees

The last module of PanToVA checks the pangenome database for any phylogenetic trees that the user might have created of the pangenome. The phylogenetic trees can be constructed by PanTools using different methods, such as core phylogeny using either Maximum likelihood (ML) or Neighbour-joining (NJ), K-mer joining using NJ, consensus tree, gene distance tree using NJ, Average Nucleotide Identity (ANI) for prokaryotic genomes or Multilocus sequence analysis (MLSA). In order to do this, PanTools uses functions from other packages and scripts. These dependencies all save the resulting trees in slightly different file formats with different extensions, which PanToVA is able to transform into formats PanVA accepts. However, these packages save the resulting phylogenetic trees in any location and under any name the user defines. To address this issue, several options were considered. For example, one option was to restrict the freedom by defining a directory in which the trees should be stored. However, this would not work for pangenomes that already exist, and for new pangenomes this would require users to be aware of the restriction before constructing a pangenome. Another option was to remove the option of naming the phylogenetic trees completely, yet often when building phylogenetic trees various settings are tweaked and adjusted to come to the right tree. This results in multiple versions of the trees, thus removing the naming option completely would make distinguishing these unnecessarily difficult. Therefore, considering PanToVA checks if all required files are present at the start of the run, it now also checks if any phylogenetic tree files are present, based on file extension. If any are found during the checks, the file locations are stored for this module where the file formats are adjusted to match the files accepted by PanVA. After which the preprocessing by PanToVA is complete and the user is presented with all files for use in PanVA.

3.8 The combined workflow

The modules described above come together to form the total workflow of PanToVA as can be seen in *Figure 6.* This workflow produces the input files for visualisation in PanVA for any PanTools pangenome. When using the command line standalone version of PanToVA, users need to update the template configuration file to match their pangenome and criteria. Subsequently, the user activates the conda environment included with PanToVA, after which the following command is used to start the preprocessing;

```
python3 path/to/pan_to_va.py path/to/config.ini
```

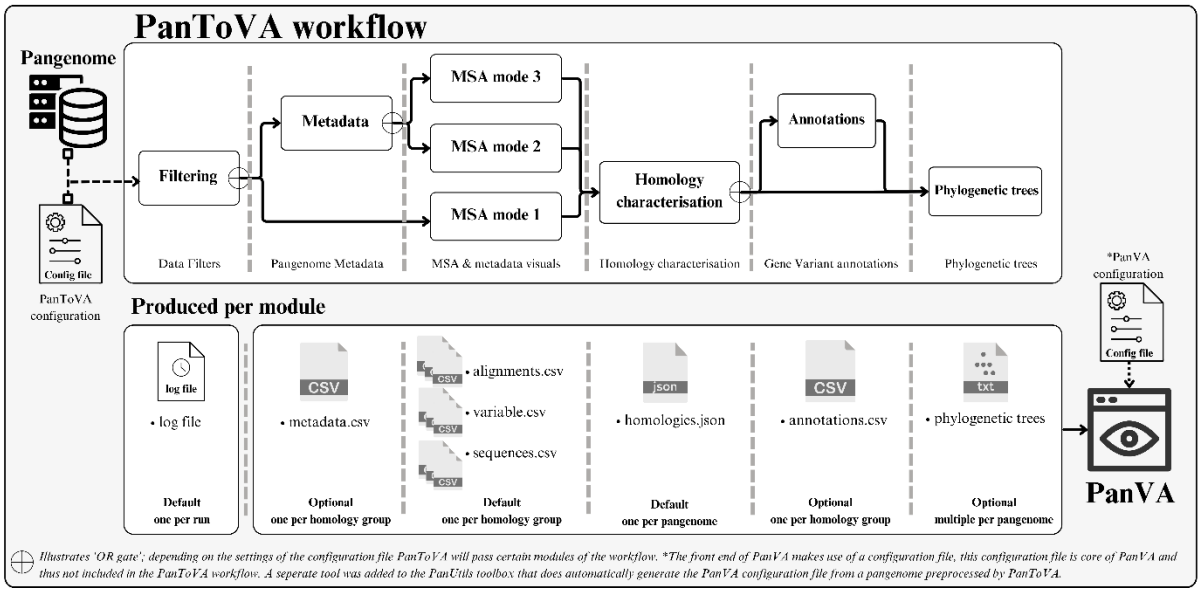


Figure 6. The complete PanToVA workflow.

4. Application

In this section, we describe how all modules work together depending on what data and settings are used. For this we used a selection of pangenomes of various origins, containing bacterial, fungal and plant pangenomes. For the bacterial pangenomes we used a *Pectobacterium* pangenome of 454 genomes (Pecto_454) with numerous species and strains with metadata but without phenotype specific elements, a smaller *Pectobacterium* pangenome consisting of 197 genomes (Pecto_197) but including both metadata and phenotype specific elements. The fungal pangenomes are represented by a Yeast pangenome consisting of 10 *Saccharomyces cerevisiae* strains. As well as two *Fusarium asiaticum* pangenomes of 60 and 245 genomes respectively. The plant pangenomes are represented by an *Arabidopsis thaliana* pangenome consisting of 25 genomes, and 15% of the homology groups of a *Lactuca* pangenome (Lettuce_15%) constructed with reference genomes of 3 species (*Lactuca sativa, Lactuca saligna* and *Lactuca virosa)* combined with 440 resequenced accessions and lastly a pangenome consisting of 20 *Solanum* genomes (Tomato_20) which includes the species; *Solanum lycopersicum, Solanum habrochaites, Solanum penelli* and *Solanum pimpinellifolium*.

The pangenomes are constructed in such a manner to demonstrate the diversity of pangenomes. This is done by including and excluding available data. *Table 1* shows the differences in pangenome size and included data. The individual preprocessing steps the aforementioned pangenomes go through is shown in *Supplementary Figure 1*.

Table 1. 8 Different pangenomes, illustrating the diversity in included data.

Pangenome ID	# genomes & # accessions	# hom groups (pre filter)	metadata classes	positional metadata	gene variant annotations	phylogenetic trees
Tomato_20	20 - 0	222	2	2	yes	kmer distance, gene distance
Pecto_454	454 - 0	27692	0	0	no	none
Pecto_197	197 - 0	996	9	1	yes	kmer distance
Arabi_25	25 - 30	880	7	1	yes	kmer distance
Yeast_10	10 - 0	2140	1	0	yes	kmer distance
Fusa_245	245 - 0	17636	15	15	no	none
Fusa_60	60 - 0	15523	15	15	no	gene distance, kmer distance, core SNP
Lettuce_15%	*3 - 440*	*4478*	*0*	*0*	*yes*	*none*

4.1 Implementation & Integration

PanToVA is created for Python v3.11.0 and makes use of existing Python (v3.11.0 compatible) packages, tools and scripts. The dependencies for PanToVA are included when installed in the form of a conda ("Anaconda Software Distribution", 2020) environment.

PanToVA is required to work together with PanTools. Therefore, the next step was integration of the preprocessing into the whole workflow of PanTools and PanVA. In order to integrate PanToVA with PanTools, the entire codebase of PanToVA is now part of the public version of PanUtils (see: **Code & Data availability**). PanUtils is a utility package for PanTools and allows users to automate the building of the pangenome. Now with the inclusion of PanToVA a user can automate the construction of the pangenome and the preprocessing for visualisation in PanVA, further improving user-friendliness by keeping all tools together as one.

PanVA requires a configuration file for the frontend of the application for every instance. This configuration file requires a substantial amount of time to prepare. In addition, due to the .json format any mistake in the structure of the file will cause it to break the instance. Therefore, as an additional feature a separate script was created that constructs a boilerplate version of the configuration file for any preprocessed pangenome. This boilerplate provides everything that is required by PanVA to work, yet allows users to adjust values for things such as headers, titles, column width and others, without having to touch the data structure itself.

4.2 Run time and optimisation

Building a sizeable pangenome takes a significant amount of time due to collecting data and running PanTools, but once constructed, PanToVA's multiprocessing approach only requires minutes for high end computer systems to a few hours for lower end specifications in relation with the pangenome size. PanToVA's run time is dependent on several factors, such as the number of assigned cpu cores, the size of the pangenome, number of included homology groups, the metadata that is included, phylogenetic trees, and if any gene variant annotations are available. Demonstrating PanToVA's ability to handle pangenomes of various orders of magnitude independent of organism, the selection of preconstructed pangenomes were preprocessed by PanToVA and timed, as shown in *Table 2*.

Table 2. PanToVA preprocessing run times per pangenome.

Pangenome ID	# genomes & # accessions	# hom groups (pre filter)	# hom groups (after filter)	filter settings [1]	metadata classes	positional metadata	gene variant annotations	phylogenetic trees	max. cpu cores [3]	run time (h/m/s/ms)
Tomato_20	20 - 0	222	222	90/2/1	2	2	yes	kmer distance,	100	00:27:48.48
Pecto_454	454 - 0	27692	20354	90/2/1	0	0	no [2]	none [2]	24	00:02:50.84
Pecto_197	197 - 0	996	908	90/2/1	9	1	yes	kmer distance	60	00:17:49.02
Arabi_25	25 - 30	880	878	90/2/1	7	1	yes	kmer distance	100	00:15:12.64
Yeast_10	10 - 0	2140	2101	90/2/1	1	0	yes	kmer distance	10	00:02:52.28
Fusa_245	245 - 0	17636	16157	90/2/1	15	15	no [2]	none	16	01:12:07.35
Fusa_60	60 - 0	15523	13296	90/2/1	15	15	no [2]	gene distance, kmer	16	00:38:27.40
Lettuce_15%*	*3 - 440*	*4478*	*2289*	*90/2/2*	*0*	*0*	*yes*	*none*	*32*	*00:10:21.15*

*1) Filter settings; minimal alignment length in bases, number of members per homology group, number of unique members per homology group. 2) Feature was not implemented at the time of preprocessing. 3) Maximum number of cpu cores PanToVA is allowed to use at once, meaning it is not neccesarrily the amount used throughout the run. *) The lettuce_15% pangenome consists of only 15% of the homology groups of the full pangenome.*

With the preprocessing of the pangenomes completed, the next step was launching the individual PanVA instances. During this step some issues were discovered with PanVA when larger pangenomes are used, specifically large homology groups with a large number of members and a long alignment length. When a PanVA instance is created and a homology group is selected for visualisation the first time, a linkage matrix is automatically calculated by PanVA to form the dendrogram for similarity. The original implementation made use of the Levenshtein (Berger et al., 2021) method for calculation of similarity. This was functional in instances that consisted of small homology groups, short alignment lengths with few members. However, for larger homology groups, combination of lots of members and long alignments, this was insufficient. For locally hosted PanVA instances the longer calculation time would be bothersome but not necessarily an issue if it would take slightly longer. However, when a PanVA instance is hosted on a dedicated server the calculation time would result in a time out from the server.

Several fixes were tested, such as increasing the time it takes for a server time out to occur. In order to test this method, one such large homology group of 443 members and an alignment length of 35947 bases was selected, this took more than 70 minutes to complete the dendrogram calculation with the old implementation, when ran locally to avoid server time outs. Thus, not only would this not fix the issue as waiting 70 minutes to view a single homology group is unacceptable, it would also pose a major security risk if hosted on a dedicated server with an adjusted server time out time to allow for this wait time.

Thus, we settled to find a more optimized method to calculate the linkage matrix. For this we compared the performance of the Levenshtein (Berger et al., 2021) method to the RapidFuzz (Bachmann, 2021) version implementation of hamming distance. From the available preprocessed pangenomes at the time, homology groups that were considered to be large homology groups were selected, in addition to homology groups of varying sizes. This was done to ensure that if differences between calculation methods existed, any correlation in run time, outcome, size or length would not be missed. As these methods both produce matrices that are used in PanVA to create the similarity matrices, the exact values of the elements in the matrices are less important than the overall differences in values across the matrix. Therefore, it is more important to check the largest and smallest differences in values. The results of this test are shown in *Table 3*, indicating that calculation times are different specifically when it comes to the larger homology groups.

The values produced by the various methods are not significantly dissimilar. In addition, the PanVA (Brandt et al., 2022) paper already referred to the hamming distance method, yet the code was using the Levenshtein method. Therefore, the RapidFuzz (Bachmann, 2021) implementation of hamming distance was implemented in PanVA.

Table 3. Linkage matrix calculation comparison. Compares the time it takes for the Hamming and Levenshtein method to generate the linkage matrix, comparing the resulting matrix on the largest and smallest differences between elements in the matrices.

Members	Alignment length	Time Levenshtein	Time Hamming	Largest difference	Smallest difference
443	11879	09:52.8	00:01.8	-1.35E-03	0.00E+00
443	11605	10:08.9	00:02.1	-1.92E-03	0.00E+00
444	12592	09:39.3	00:03.4	-1.24E-03	0.00E+00
442	22962	43:42.3	00:04.5	-2.25E-03	0.00E+00
5	10935	00:00.0	00:00.1	-6.80E-03	0.00E+00
3	7854	00:00.0	00:00.0	-3.61E-03	0.00E+00
6	301	00:00.0	00:00.0	-1.88E-04	3.29E-03
5	10935	00:00.0	00:00.0	-6.80E-03	0.00E+00
5	3641	00:00.0	00:00.0	-2.50E-03	0.00E+00
5	6748	00:00.0	00:00.0	-3.51E-04	0.00E+00
12	168	00:00.0	00:00.0	-4.99E-04	5.92E-03
7	1233	00:00.0	00:00.0	-5.73E-05	0.00E+00
442	1326	00:08.1	00:00.9	-3.42E-05	7.54E-04
8	839	00:00.0	00:00.0	-5.63E-03	0.00E+00
19	546	00:00.0	00:00.0	-5.79E-02	1.45E-03
12	306	00:00.0	00:00.0	-5.68E-04	0.00E+00
445	3199	00:38.8	00:02.0	-3.33E-04	0.00E+00
1330	1985	03:44.5	00:09.1	-3.11E-03	0.00E+00
2	13192	00:00.0	00:00.0	-2.04E-03	0.00E+00
2	11867	00:00.0	00:00.0	-4.42E-04	0.00E+00
442	492	00:02.1	00:01.5	-4.41E-03	0.00E+00
2	492	00:00.0	00:00.0	0.00E+00	1.87E-03
2	7215	00:00.0	00:00.0	-3.08E-03	0.00E+00

Members) number of members in the homology group. Alignment length) the length of the multiple sequence alignment in number of bases. Time Levenshtein) run time of the levenshtein method of calculation. Time Hamming) run time of the hamming method of calculation. Largest difference) the largest difference found between values of the two matrices. Smallest difference) the smallest difference found between values of the two matrices.

4.3 Resulting PanVA instance

Each of the aforementioned pangenomes made it through the preprocessing without issues. For the last test to ensure PanToVA was working as intended, for each of the different pangenomes a PanVA instance was made, with the exception of the complete Pecto_454 pangenome due to time limitations. The *Figure 7* shows a schematic still of the Yeast_10 pangenome PanVA instance, highlighting the different elements of the PanVA instance and which input files are used for what sections in the instance. PanVA is an interactive tool for visual exploration of pangenomes, so its features are best experienced by interacting with the tool itself. Therefore, a selection of the PanVA instances created for this project are available to the public (**Code & Data availability**).

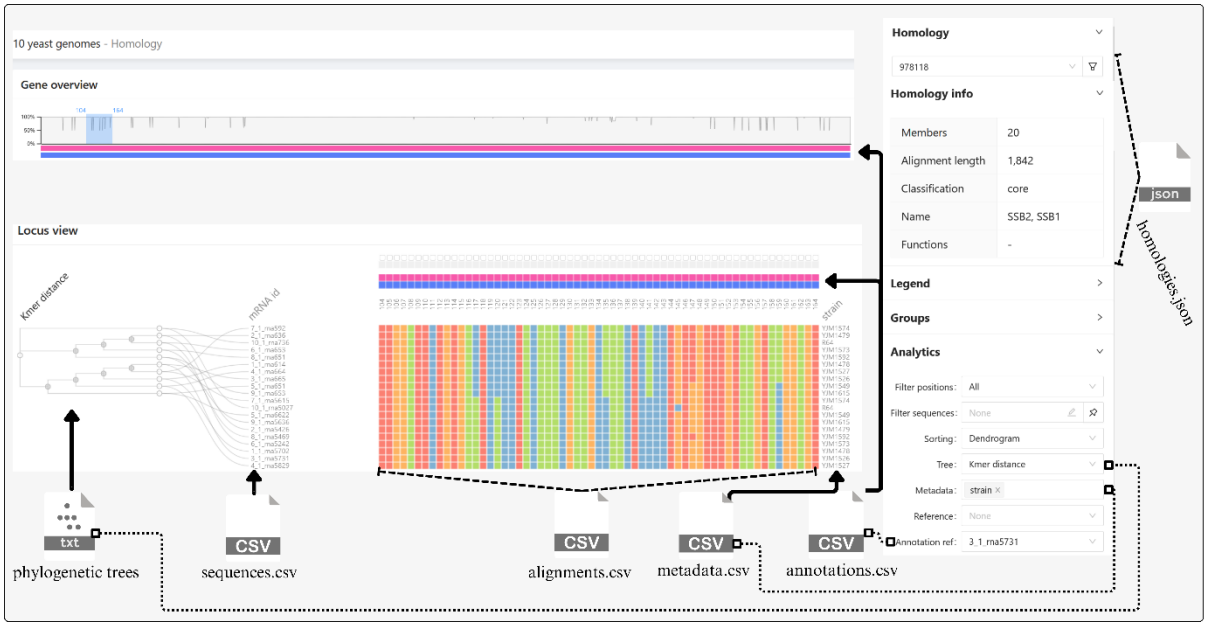


Figure 7. Schematic PanVA instance of Yeast_10 homology group 978118. Highlighting which features of PanVA are affected by the files created by PanToVA to visualise pangenomes. Filled and dashed lines indicate which files store the data to produce what segment. Dotted lines represent how and where users of PanVA can interact with the data during visualisation.

5. Discussion

PanToVA requirements were set to be efficient, user-friendly, configurable, applicable for all PanTools pangenomes and automated whilst taking into consideration the needs and preferences of the assumed user base. Even with PanToVA meeting the and in cases exceeding the requirements, there were still some limitations as to what could be achieved, as some challenges arose throughout the project.

One such challenge were the frequent and irregularly scheduled updates to both PanTools and PanVA. As PanToVA is to facilitate the preprocessing of data between these tools, the updates resulted in several rewrites and adjustments to the PanToVA codebase. This, in combination with the changing requirements for PanToVA and the limited duration of the project, resulted in some level of redundancy built into the design of PanToVA. Initially this was viewed as a flaw, however, one such redundancy became an important asset of PanToVA. Considering reconstruction of pangenomes for every minor update of PanTools was not feasible within the project time schedule, this redundancy was created in order to catch any minor differences in file formats that can be present depending on which (developmental) version of PanTools was used to build the pangenome. Because of this redundancy, PanToVA is now backwards compatible with pangenomes created using older versions of PanTools, which allows users to use already existing pangenomes.

Another challenge arose, as during the PanToVA project the functionalities of PanVA were expanded, which enabled it to visualise more data than originally planned. As a result, the project schedule for PanToVA was adjusted to accommodate these changes. This resulted in an overall more robust and able PanToVA, yet due to the shifting priorities there was less time for the phylogenetic tree module to be optimised.

The phylogenetic tree module will search for any files in the pangenome that might be a phylogenetic tree file, as neither file extension, file name and subdirectory storage location is standardised in PanTools. Several methods to improve the phylogenetic tree module have been discussed. The first option being standardising the storage location of these phylogenetic trees in the pangenome by PanTools, because this would make searching pangenome as done by PanToVA obsolete. This would

reduce run time of PanToVA even further, especially for larger pangenomes as less files need to be searched. Another option would be to let users specify which phylogenetic tree(s) they want to add for visualisation in the configuration file. However, we feel this would not be as efficient or as user friendly seeing as it is a common practice when constructing phylogenetic trees to make several versions using different methods. Therefore, we argue having users enter several filenames and locations in order to include all phylogenetic trees would negatively impact user friendliness. These options are merely suggestions and have been passed on to the development team of PanTools and PanVA with advice on how to implement either option in PanToVA as well.

Throughout the project several small improvements, for cases such as standardising naming convention, file extensions or in file data structures, in collaboration with the developers of both PanTools and PanVA, proved to be successful and resulted in a more efficient workflow. Thanks to this collaboration, it also became apparent that some of the choices made for PanVA that were beneficial for visualisation, placed unnecessary restrictions on the available data, thus limiting PanToVA. For example, PanToVA currently removes any homology group that has no variants. As by design PanVA is a visual variant analysis tool for pangenomes on a homology group level, and therefore it won't allow the users to include any homology groups that have no variation. Which means removing a sometimes sizable number of homology groups that contain highly conserved genes. This is less important when it comes to known genes that are present in each of the genomes of the pangenome, yet when it comes to highly conserved genes that are only found in a specific set of genomes within the pangenome this might not be so trivial. Therefore, recommendations have been made to the PanVA developers to allow for visualisation of non-variant homology groups.

A similar limitation is due to PanToVA's requirement for trimmed multiple sequence alignments of the homology groups. This requirement was known from the start of the project as PanVA was only able to work for trimmed alignments. However, during the later stages of the project it became apparent that some pangenomes, especially for pangenomes with limited number of genomes but of higher plants, PanTools was not always able to provide trimmed alignments. This can be due to the protein sequences starting and ending on the same position or the resulting trimmed alignment being to small. Thus, the request was made to alter the preprocessing to allow for untrimmed alignments, however, when work started on this request it was discovered that PanVA itself was unable to work with untrimmed alignments. While inspecting the PanVA codebase to resolve this issue, we found that part of the problem was due to PanVA expecting to find the label "nuc_trimmed_seq" in several locations. It would have taken too long to alter this in both PanVA and PanToVA in the remaining time of the PanToVA project without guaranteeing it would completely resolve the issue. Thus, the suggestion was made to change this during a follow up project after PanVA is updated and able to handle these untrimmed alignments.

In addition, the annotation options of PanTools did not come into the release branch of PanTools until the later stages of the PanToVA project. Therefore, the PanToVA implementation for the annotations was done in a limited timeframe, and therefore did not receive the same level of tests to ensure optimised functioning. However, the late introduction of this module does highlight the success of the modular design, demonstrating modules can be modified, removed and added without interrupting the other modules.

Reflecting back on the PanToVA project as a whole, we reiterate the importance of proper communication and checkpoints, especially when collaborating with developers across different projects. As this project was heavily tied into the PanTools and PanVA projects, changes that occurred in one of the projects could have an effect on all projects. For example, postponed updates, moved datasets, and rescheduled deadlines sometimes negatively impacted the PanToVA project. However we quickly adapted and improved the lines of communication between the parties involved. This resulted in fewer issues in compatibility of the tools, and increased understanding of the combined codework as

a whole which allowed us to suggest and make improvements to PanTools and PanVA outside of the PanToVA project boundaries. In the future, we recommend taking some time at the start of the project to define a clear communication strategy with the key collaborators.

As for the choice for Python, other languages could have been used, and specifically java could have been a good option. Seeing as PanTools is mostly written in java this could have improved integration in the long run. But as Python is considered to be more readable, versatile and has simpler syntax and the PanToVA project aims to be widely applicable and simple, using Python is a good option. In addition, the preprocessing by PanToVA is predominantly parsing, data collection, formatting and filtering and does not require heavy calculations to be done. Had this been the case arguments could have been made to use other options such as R or Java.

With PanToVA we streamlined the workflow of pangenome exploration through visualisation. Retrieving data of interest from the large and complex data structures of pangenomes, constructing files from the pangenome built with PanTools without the need for manual curation. Providing PanVA the data which allows it to visualise multiple references together with providing the adequate context of heterogeneous (meta)data, such as annotations, evolutionary relationships and phenotypes (Brandt et al., 2022). This further improves the ease in which genetic variation underlying phenotypes can be explored. PanToVA, together with PanTools and PanVA, brings us one step closer to fully transitioning from the single reference genome to the pangenome approach, allowing research to look at variation on a larger scale.

Conclusion

During this project, we have developed a Python workflow that is able to automatically preprocess PanTools pangenomes for visualisation in PanVA. This workflow works as a standalone tool and is also included in PanUtils under the “export-to-panva” project. This project sought to create a workflow that works efficiently for all pangenomes and is both automated and configurable. From the resulting applications, test runs and the run times we conclude that PanToVA works efficiently for both small and larger pangenomes. We have also shown that PanToVA is usable for pangenomes of various organisms and the different contents of the pangenomes. In addition, we have shown the usefulness of PanToVA as it provides users with insights of what occurs during the preprocessing improving the reproducibility of any study using PanVA. Following this, the in-house user base was able to work well with the application solely relying on the instructions of the configuration file, highlighting the ease of use. The ease of use was further confirmed during workshops with external potential users of the workflow. From the application itself we can conclude that PanToVA is modular and expandable, as modules can be activated independently, and additional modules can be added as shown by the incorporation of the annotation module during the latest stages of the project. In conclusion, PanToVA has met and exceeded the project requirements and is incorporated in the pangenome variant analysis workflow. As a final thought, based on the findings and the gained experiences of the project we recommend a follow up project to polish the details of the PanToVA project in a collaboration project with the developers of PanTools and PanVA.

Code & Data availability

Parties interested in the possibilities of setting up the presented pangenome variant analysis workflow at their facilities, please contact S. Smit from the Bioinformatics Group of Wageningen University & Research.

The codebase of **PanToVA** is publicly available as part of the **PanUtils** package at https://github.com/PanUtils/export-to-panva.

The data used in the creation of the *Pectobacterium, Fusarium, Tomato, Lactuca & Solanum* pangenomes and PanVA instances, are available from Wageningen University and Research – Bioinformatics Group. Restrictions apply to the availability of these datasets, which are under license for this study. None of the data used for the creation of the pangenomes and the PanVA instances is my own. Some of the aforementioned datasets might become available at a later date, after embargo period following publication of papers on the data. Data requests should be addressed to dr. S. Smit, of the Bioinformatics Group.

The resulting *Saccharomyces* and *Arabidopsis* and the small *Pectobacterium* PanVA instances are publicly available at www.bioinformatics.nl/panva/yeast10/, www.bioinformatics.nl/panva/ara25/, www.bioinformatics.nl/panva/pecto197/, using the guest access; "workshop", "panvatest!". These instances are actively used to test new features of PanVA, thus are known to occasionally produce errors and are subject to change.

PanTools available at (recommended) https://anaconda.org/bioconda/pantools or over at https://git.wur.nl/bioinformatics/pantools (developers).

PanVA code available at https://github.com/PanBrowse/PanVA.

Figures are made using the free version of Canva; https://www.canva.com/

Acknowledgements

I would like to thank dr. S. Smit (WUR, Bioinformatics Group) and MSc. M. Yang (WUR, Bioinformatics Group) for their invaluable supervision, support and time during the project. In addition, we thank A. van den Brandt (WUR, Bioinformatics Group) and Folkert de Vries for their input and time throughout the project. We also want to thank R. van Esch (WUR, Bioinformatics Group) and D. van Workum (WUR, Bioinformatics Group) for their help with the integration of PanToVA with PanTools and PanVA. Additionally, we thank L. Pardeshi (WUR, Bioinformatics Group) for his role as supervisor in the early stages of the project. Lastly, we thank everyone from the Bioinformatics Group who helped make this project possible by providing useful insights and a great working environment.

Bibliography

Anaconda Software Distribution. (2020). In *Anaconda Documentation*. Anaconda Inc. https://docs.anaconda.com/

Bachmann, M. (2021). *GitHub - maxbachmann/RapidFuzz: Rapid fuzzy string matching in Python using various string metrics*. https://github.com/maxbachmann/RapidFuzz

Ballouz, S., Dobin, A., & Gillis, J. A. (2019). Is it time to change the reference genome? *Genome Biology*, *20*(1), 1–9. https://doi.org/10.1186/S13059-019-1774-4/FIGURES/3

Berger, B., Waterman, M. S., & William Yu, Y. (2021). Levenshtein Distance, Sequence Comparison and Biological Database Search HHS Public Access. *IEEE Trans Inf Theory*, *67*(6), 3287–3294. https://doi.org/10.1109/tit.2020.2996543

Brandt, A. Van Den, Jonkheer, E. M., Workum, D. M. Van, Wetering, H. Van De, Smit, S., & Vilanova, A. (2022). *PanVA : Variant Analysis within Pangenomes*. *14*(8), 1–14.

Clegg, D., & Barker, R. (1994). CASE method fast-track - a RAD approach. In *undefined*. Pearson Education Limited.

Hudson, T. J., Anderson, W., Aretz, A., Barker, A. D., Bell, C., Bernabé, R. R., Bhan, M. K., Calvo, F., Eerola, I., Gerhard, D. S., Guttmacher, A., Guyer, M., Hemsley, F. M., Jennings, J. L., Kerr, D., Klatt, P., Kolar, P., Kusuda, J., Lane, D. P., … Wainwright, B. J. (2010). International network of

cancer genome projects. *Nature*, *464*(7291), 993–998. https://doi.org/10.1038/nature08987

Jonkheer, E. M., van Workum, D. J. M., Sheikhizadeh Anari, S., Brankovics, B., de Haan, J. R., Berke, L., van der Lee, T. A. J., de Ridder, D., & Smit, S. (2022). PanTools v3: functional annotation, classification and phylogenomics. *Bioinformatics*, *38*(18), 4403–4405. https://doi.org/10.1093/BIOINFORMATICS/BTAC506

Katoh, K., & Standley, D. M. (2016). *A simple method to control over-alignment in the MAFFT multiple sequence alignment program*. https://doi.org/10.1093/bioinformatics/btw108

Li, Z., Fu, B. Y., Gao, Y. M., Wang, W. S., Xu, J. L., Zhang, F., Zhao, X. Q., Zheng, T. Q., Zhou, Y. L., Zhang, G., Tai, S., Xu, J., Hu, W., Yang, M., Niu, Y., Wang, M., Li, Y., Bian, L., Han, X., … Leung, H. (2014). The 3,000 rice genomes project. *GigaScience*, *3*(1), 1–6. https://doi.org/10.1186/2047-217X-3-7/FIGURES/2

Marroni, F., Pinosio, S., & Morgante, M. (2014). Structural variation and genome complexity: Is dispensable really dispensable? *Current Opinion in Plant Biology*, *18*(1), 31–36. https://doi.org/10.1016/J.PBI.2014.01.003

Morgante, M., De Paoli, E., & Radovic, S. (2007). Transposable elements and the plant pan-genomes. *Current Opinion in Plant Biology*, *10*(2), 149–155. https://doi.org/10.1016/J.PBI.2007.02.001

Muir, P., Li, S., Lou, S., Wang, D., Spakowicz, D. J., Salichos, L., Zhang, J., Weinstock, G. M., Isaacs, F., Rozowsky, J., & Gerstein, M. (2016). The real cost of sequencing: Scaling computation to keep pace with data generation. *Genome Biology*, *17*(1). https://doi.org/10.1186/S13059-016-0917-0

National Human Genome Research Institute. (2020). *NOT-OD-21-013: Final NIH Policy for Data Management and Sharing*. https://grants.nih.gov/grants/guide/notice-files/NOT-OD-21-013.html

Nuin, P. A. S., Wang, Z., & Tillier, E. R. M. (2006). The accuracy of several multiple sequence alignment programs for proteins. *BMC Bioinformatics*, *7*(1), 1–18. https://doi.org/10.1186/1471-2105-7-471/TABLES/1

Nusrat, S., Harbig, T., & Gehlenborg, N. (2019). Tasks, techniques, and tools for genomic data visualization. *Computer Graphics Forum*, *38*(3), 781–805. https://doi.org/10.1111/CGF.13727

Pedersen, T. L., Nookaew, I., Wayne Ussery, D., & Månsson, M. (2017). PanViz: Interactive visualization of the structure of functionally annotated pangenomes. *Bioinformatics*, *33*(7), 1081–1082. https://doi.org/10.1093/BIOINFORMATICS/BTW761

Ranwez, V., & Chantret, N. N. (2020). *Strengths and Limits of Multiple Sequence Alignment and Filtering Methods*. https://hal.inria.fr/PGE.

Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, J. C., Hutchison, C. A., Slocombe, P. M., & Smith, M. (1977). Nucleotide sequence of bacteriophage φX174 DNA. *Nature 1977 265:5596*, *265*(5596), 687–695. https://doi.org/10.1038/265687a0

Sheikhizadeh, S., Schranz, M. E., Akdel, M., De Ridder, D., & Smit, S. (2016). PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*, *32*(17), i487–i493. https://doi.org/10.1093/BIOINFORMATICS/BTW455

Sherman, R. M., & Salzberg, S. L. (2020). Pan-genomics in the human genome era. *Nature Reviews Genetics 2020 21:4*, *21*(4), 243–254. https://doi.org/10.1038/s41576-020-0210-7

Staden, R. (1982). Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing. *Nucleic Acids Research*, *10*(15), 4731–4751. https://doi.org/10.1093/NAR/10.15.4731

Tettelin, H., Masignani, V., Cieslewicz, M. J., Donati, C., Medini, D., Ward, N. L., Angiuoli, S. V., Crabtree, J., Jones, A. L., Durkin, A. S., DeBoy, R. T., Davidsen, T. M., Mora, M., Scarselli, M., Margarit Y Ros, I., Peterson, J. D., Hauser, C. R., Sundaram, J. P., Nelson, W. C., … Fraser, C.

M. (2005). Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: Implications for the microbial "pan-genome." *Proceedings of the National Academy of Sciences of the United States of America*, *102*(39), 13950–13955. https://doi.org/10.1073/PNAS.0506758102

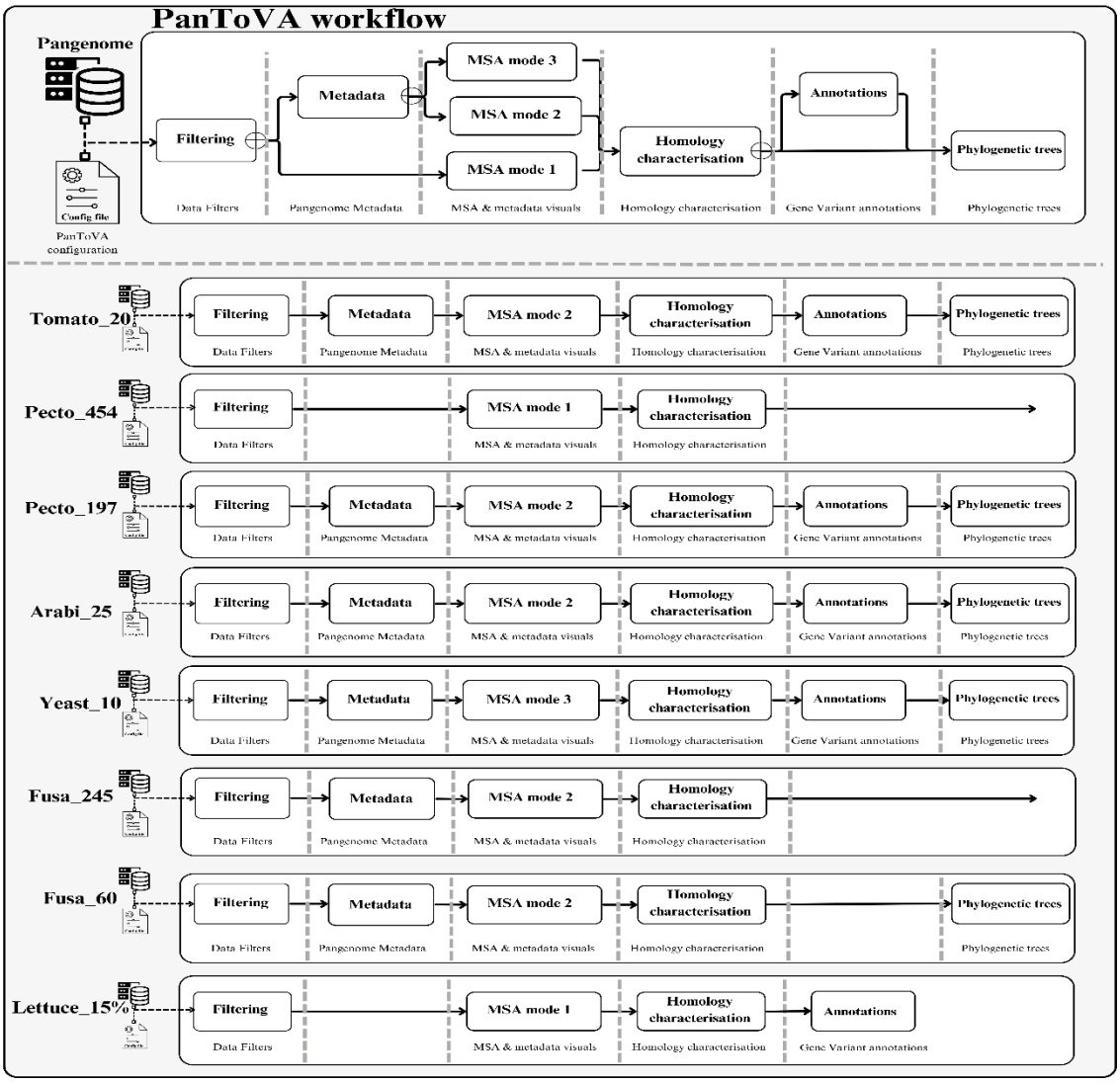
the PanTools team. (2023). *Multiple Sequence Alignments — PanTools 4.2.0 documentation*. https://pantools.readthedocs.io/en/v4.2.0/user_guide/msa.html

Vernikos, G., Medini, D., Riley, D. R., & Tettelin, H. (2015). Ten years of pan-genome analyses. *Current Opinion in Microbiology*, *23*, 148–154. https://doi.org/10.1016/J.MIB.2014.11.016

Wang, S., Qian, Y.-Q., Zhao, R.-P., Chen, L.-L., & Song, J.-M. (2023). Graph-based pan-genomes: increased opportunities in plant genomics. *Journal of Experimental Botany*, *74*(1), 24–39. https://doi.org/10.1093/JXB/ERAC412

Zhang, J., Chen, L. L., Xing, F., Kudrna, D. A., Yao, W., Copetti, D., Mu, T., Li, W., Song, J. M., Xie, W., Lee, S., Talag, J., Shao, L., An, Y., Zhang, C. L., Ouyang, Y., Sun, S., Jiao, W. B., Lv, F., … Zhang, Q. (2016). Extensive sequence divergence between the reference genomes of two elite indica rice varieties Zhenshan 97 and Minghui 63. *Proceedings of the National Academy of Sciences of the United States of America*, *113*(35), E5163–E5171. https://doi.org/10.1073/PNAS.1611012113/-/DCSUPPLEMENTAL/PNAS.1611012113.SD01.XLSX

Supplementary



Supplementary Figure 1. PanToVA workflow routes per pangenome.