Bioinformatics Program

Technical University of Munich

Ludwig-Maximilians-Universität München

Bachelor's Thesis in Bioinformatics

Bioinformatic Methods of Individual and Pangenome Variations

Lukas Jaeger

Bioinformatics Program

Technical University of Munich

Ludwig-Maximilians-Universität München

Bachelor's Thesis in Bioinformatics

Bioinformatic Methods of Individual and Pangenome Variations

Bioinformatische Methoden für Individuelle und Pan-genomische Variationen

Author:	Lukas Jaeger
Supervisor:	Prof. Dr. Ralf Zimmer
Advisors:	Armin Hadziahmetovic, Samuel Klein
Submitted:	15.08.2023

Statement of Originality

I confirm that this Bachelor's thesis with the title

Bioinformatic Methods of Individual and Pangenome Variations

Bioinformatische Methoden für Individuelle und Pan-genomische Variationen

is my own work and I have documented all sources and material used.

_____ Munich, the August 2023

Abstract

A pangenome combines the genomes of several individuals of a species. This allows for better coverage of genomic diversity in the population and a better understanding of gene-disease associations. In 2022, the Human Pangenome Reference Consortium published a new version of the human pangenome containing a total of 44 haplotype-reserved genome assemblies of individuals and the two linear reference genomes GRCh38 and CHM13. This pangenome was published as a text-based GFA file. In this paper, the general structure of this file is analyzed in order to better understand and use it. In addition, the pan2 workflow, consisting of the three tools pan2prep, pan2index and pan2gene, will be presented as a new method to extract the different gene sequences from the pangenome and thus make the data from the pangenome more accessible. For each gene, a separate GFA file is written. In addition, a VCF file with the genetic variants, a Fasta file with the haplotype sequences and an MSA are returned. However, this method has some limitations and offers room for improvement. The analysis of the pangenome file has shown that the data is not stored optimally and that there is room for improvement here as well. The extracted data can then be integrated into ISAR to provide a better understanding of the diversity and variability of the human genome.

In einem Pangenom sind die Genome mehrerer Individuen einer Spezies zusammengefasst. Dies ermöglicht eine bessere Erfassung der genomischen Diversität in der Bevölkerung und ein besseres Verständnis von der Beziehung zwischen Genen und Krankheiten. Im Jahr 2022 veröffentlichte das Human Pangenome Reference Consortium eine neue Version des menschlichen Pangenoms, die insgesamt 44 nach Haplotypen unterteilte Genom-Assemblierungen von Individuen sowie die beiden linearen Referenzgenome GRCh38 und CHM13 enthält. Dieses Pangenom wurde als textbasierte GFA-Datei veröffentlicht. In dieser Arbeit wird die allgemeine Struktur dieser Datei analysiert, um sie besser zu verstehen und handhaben zu können. Darüber hinaus wird mit dem pan2-Workflow, bestehend aus den drei Tools pan2prep, pan2index und pan2gene, eine neue Methode vorgestellt, um die verschiedenen Gensequenzen aus dem Pangenom zu extrahieren und damit die Daten aus dem Pangenom allgemein zugänglicher zu machen. Für jedes Gen wird dann eine eigene GFA-Datei geschrieben. Zusätzlich wird eine VCF-Datei mit den Varianten, eine Fasta-Datei mit den Haplotypsequenzen und ein MSA ausgegeben. Diese Methode hat jedoch einige Einschränkungen und bietet Raum für Verbesserungen. Die Analyse der Pangenom-Datei hat gezeigt, dass die Daten nicht optimal gespeichert sind und dass es auch hier noch Verbesserungsmöglichkeiten gibt. Die extrahierten Daten können dann in ISAR integriert werden, um ein besseres Verständnis der Diversität und Variabilität des menschlichen Genoms zu ermöglichen.

Acknowledgments

I want to thank Prof. Dr. Ralf Zimmer, Armin Hadziahmetovic and Samuel Klein for their guidance for the project of writing this thesis. I also want to thank Katharina Schmid for proofreading.
Additionally, I would like to thank my family and friends for their support during this stressful time.

Contents

List of Figures

List of Tables

1 Introduction

Since the complete human genome was sequenced in 2001, human DNA has been studied in ever greater detail [1]. Understanding genetics is important to understand genetic disease. For example, aside from accidents, genomic factors contribute to nine out of the ten leading causes of death in the United States, including diseases such as heart disease, cancer, and diabetes [2]. As a fundamental resource, reference genomes were used, which were increasingly optimized by closing gaps and correcting errors. They were assembled from sequence data of different human individuals and symbolize the human genome as a whole [3]. However, human genomes are only about 99.9% identical [2]. Yet the remaining 0.1% can be decisive for susceptibility to certain diseases. For example, it has been shown that one in five people who did not develop symptoms during a SARS-CoV-2 infection have a particular gene variant called HLA-B*15:01. People with this gene variant also have an eight times lower likelihood of developing symptoms [4]. A linear reference genome does not sufficiently reflect the diversity of the human population to be able to conduct such studies efficiently and on a large scale. Furthermore, several of the early genome-wide association study efforts focused primarily on individuals of European ancestry, limiting overall detection power and the generalizability of results to individuals of other ancestries [5].

In 2022 the Human Pangenome Reference Consortium (HPRC) released a human pangenome [6]. A pangenome describes a combination of several genomes of the same species. In this case, the pangenome is put into a graph structure, onto which reads can be mapped as with a linear reference genome. But the difference is that in a pangenome, human genomic diversity is better represented and more genetic variants are included. Thereby, it is intended to improve gene-disease association studies between different populations and to develop a better understanding of the human genome in general. [6]

New tools and methods need to be developed to work with a complex pangenome rather than a linear reference genome. A pangenome can be represented in different formats, such as a graph structure as in HPRC, but also in a database as in gnomAD [7]. Tools such as vg or odgi have already been established for processing graph-like pangenomes [8, 9]. These enable reads to be mapped onto the pangenome, multiple genome assemblies to be merged into a pangenome graph, variants to be extracted from the graph, and the graph structure to be visualized. One function that is not yet available, however, is the extraction of all haplotypes of a gene contained in the pangenome. Therefore, this thesis presents the newly developed pan2 workflow, which closes this gap.

In addition, the structure of the HPRC pangenome file is not trivial to understand and requires a certain amount of work to be able to handle it well. Therefore, the objective of this thesis can be divided into two parts: Firstly, a new tool is developed

that is capable of extracting genetic data from the HPRC pangenome file. Secondly, the general structure of the file is to be decoded and analyzed in order to make it easier to use for future studies.

For this purpose, the used material is explained in more detail in Chapter 2. Chapter 3 then describes the methods used by pan2 to extract gene data from the pangenome. An analysis of the data generated by the tool, as well as the pangenome file itself, is presented in Chapter 4, followed by some alternative approaches to using a pangenome in Chapter 5.

2 Materials

2.1 Reference Genome

The development of the human reference genome has been a significant milestone in genomics research, laying the foundation for all clinical, comparative, and population genomic analyses. Today the human reference genome is one of the most used resource in human genetics [6, 10]. The initial reference genome draft was released in 2001 [1]. This draft comprised a physical map containing more than 96 % of the euchromatic part of the human genome was combined with additional sequences from public databases. This made it the largest genome to be sequenced at this point, and the first vertebrate genome sequenced to this extent. [1]

It marked a breakthrough in genetic research and served as the foundation for biomedical and human genetic research up until today. Since then, the reference genome has served as a common frame of reference for the identification and localization of genetic variants in individual samples, and has become a fundamental prerequisite of most comparative genetic analyses [10]. Therefore, continuous efforts have been made to complete all missing parts of the reference genome and to further improve its accuracy, resulting in the two currently established reference genomes GRCh38 and CHM13 [11].

2.1.1 GRCh38

The reference genome GRCh38 was released by the Genome Reference Consortium (GRC) in 2013 and has served as the gold standard of sequence-based reference genomes (until the release of the CHM13 in 2022) due to its relatively high base-pair accuracy, and robust representations of repetitive and segmentally duplicated genomic regions. This was achieved via a clone-based assembly approach and Sanger sequencing methods: [3] GRCh38 contains the merged haplotypes of over 20 individuals, with genetic clones from a single donor forming the majority (approximately 70 %) of the sequence [6]. It includes alternate loci scaffolds that provide multiallelic and multihaplotypic representation for regions across the genome. This allowed to create a panhuman genome which accounts for genetic diversity throughout the population rather than the genome of a single person or a limited group. It contains sequences for all chromosomes and was once praised as the "most complete and accurate representation of the human genome yet produced". [3]

However, several problems limit the usefulness of the GRCh38: Firstly, numerous challenging and complex regions of the reference genome have remained unresolved, exhibiting issues such as collapsed duplications or missing sequences [10]. Secondly, only the

euchromatic regions in the genome are covered by GRCh38 while the heterochromatic ones are missing. It can therefore account for just 92 % of the human genome [11]. Third, the merging of over 20 different haplotypes resulted in reference biases [6] as well as assembly gaps that were unresolvable with BAC-based assembly techniques due to incompatible structural polymorphisms. Other repetitive and polymorphic regions are incomplete or contain assembly errors. [11]

2.1.2 CHM13

These issues were fixed with the release of genome CHM13 ("uniformly homozygous CHM13hTERT cell line") by the Telomere-to-Telomere (T2T) Consortium in 2022: This human reference genome contains a complete 3.055 billion base pair sequence, adding roughly 200 million new base pairs of sequence compared to the GRCh38. A total of 63.494 genes are identified within CHM13. Additionally, many structural errors of its predecessor were fixed. This makes T2T-CHM13 the first gap-less human reference genome, covering all chromosomes except Y. It includes all heterochromatic regions. This was achieved using long-read complementary shotgun sequencing techniques to overcome the difficulties of BAC-based assembly with polymorphism between genomes, namely Pacific Biosciences (PacBio) High Fidelity (HiFi) reads combined with Oxford Nanopore Technology (ONT). A single, complete, hydatidiform and nearly homozygous mole (CHM) was sequenced. [11]

2.2 HPRC Pangenome

The human reference genome is a central component in most studies of human genetics [6]. Despite multihaplotypic representation in GRCh38 for some regions, it includes only one haplotype for each position, creating reference biases, and thus cannot possibly represent human diversity. To circumvent this problem, the Human Pangenome Reference Consortium (HPRC) has sought a more modern, comprehensive representation of the human genome: the pangenome [6]. The goal of the HPRC is to combine the genomes of 1000 samples in the pangenome to represent as many variants of the human genome as possible. These samples are provided by the 1000 Genomes Project (1KGP), which includes data from 26 populations [12], and by the BioMe Biobank at Mount Sinai as well as a cohort of African American individuals recruited by Washington University [6]. In 2022, the current version of the pangenome was published, which contains two respective haplotypes of a total of 44 samples, as well as the two reference genomes GRCh38 and CHM13, resulting in a total of 90 genomes. This data is merged into a graph structure. The graph construction and thus a multiple sequence alignment of whole genome assemblies is highly complex, but possible due to recent improvements in tools like minimap2, cactus or pggb of which cactus and pggb construct a more accurate graph with more variants [13]. The pangenome used in this thesis was constructed by the pggb pipeline: For this, pairwise sequence alignments with wfmash are performed first. Then the graph induction with seqwish is executed and finally the graph is normalized by smoothxg. [14]

The pangenome is stored in the so-called GFA format (Graphical Fragment Assembly) [15]. GFA files contain information about DNA fragments, their connections and

the underlying sequencing data. It is ideal for representing pangenomes due to its flexibility and ability to represent complex genomic structures. In addition, the GFA format allows for the integration of different sequencing technologies. Pangenome studies often involve different sequencing approaches, such as short-read sequencing and long-read sequencing. Conveniently, there is also already a number of tools for processing and visualizing GFA files, e.g. vg or odgi [8, 9]. There are two versions of GFA files: GFA 1 and GFA 2, which extends the GFA 1 format with additional options. The pangenome used in this thesis is saved the GFA 1 type.

GFA is a tab-delimited text format that contains a set of sequences and the information of their different arrangements. The basic idea of the format is to interpret the pangenome as a directed graph that can split and reassemble. Thereby, each node contains a DNA sequence. If the node is always passed through when the graph is traversed, it is a conserved sequence, which is identical in each of the sample genomes at that position. If the graph splits, it is a variant.

Besides the header line, which often contains the version of the GFA specification, there are three different types of lines in the GFA format: Segments, links and paths. Segments contain a DNA sequence in addition to a unique segment id. They reflect the nodes in the graph. Links correspond to the edges in the graph. They link the individual segments together based on their ids and thus form the structure of the pangenome. The Paths contain the sequencing data of the samples. They consist of a sequence of consecutive segment IDs, each of which is marked with a direction char '+' or '-'. '-' means here that the reverse complement of the segment is meant, which occurs particularly with reads from the opposite strand. Here a short example for a GFA structure is provided:

H	VN:1.0				
S	seg1	ATCCGC			
S	seg2	T			
S	seg3	A			
S	seg4	TTCCGATG			
P	path1	seg1+,seg2+,seg4+			*
P	path2	seg1+,seg3+,seg4+			*
L	seg1	+	seg2	+	*
L	seg1	+	seg3	+	*
L	seg2	+	seg4	+	*
L	seg3	+	seg4	+	*

In this case there are four segments, two paths and four links. A segment line has 3 entries, a path line 4 and a link line 6. Even without the header flag of a line type (S,P,L), the different types can be identified based on their structure. The segment ids are seg1, seg2, seg3 and seg4, and the path ids are path1 and path 2. A variant is created by the bubble in the graph at seg2 and seg3. A path contains sequencing data of a sample. Here, sample 1 has a T at the seventh position, while sample 2 has an A at this position. This means that there is a genetic variant between the samples at this position. Note that not every possible traversal through the graph must necessarily be mapped in a path. Some variations do not occur simultaneously in any of the

organisms. The characters + and ∗ in the path and link lines indicate the direction of the connection of the segments and the size of an overlap. However, they are not relevant for the further course of this thesis.

In the case of the pggb HPRC pangenome used [6], further specifications can be made. The segment ids are always strings. However, here they are ascending integers converted to strings. The first segment id is therefore 1 and the thousandth is 1000. A path id looks like this:

$$\underbrace{\text{HG00438\#1}}_{\text{sample id}}\#\underbrace{\text{JAHBCB010000006.1}}_{\text{contig id}}$$

Each sample, except the two reference genomes, is divided into contigs. A path stands for a contig. The sample shown here is referenced with HG00438. , As the human is a diploid organism, the number directly behind specifies the haplotype of the sample, that is, whether it is maternal or paternal. In the context of this work, both haplotypes are analyzed independently so that the sample id is defined in the following as the concatentation of the sample and the haplotype. The second part of the String is the genebank id of the contig sequences. It will get referred to as contig id. The paths in the pangenome GFA file are sorted in ascending order by the integer suffix of the sample id and then by the suffix of the contig id. Last is the CHM13 and then the GRCh38 path. Overall, the file is additionally subdivided into several chromosomes. Each chromosome has first its segment block, then its path block and finally its link block. Overall the pangenome GFA file is approximately 95 GB large. It is still about 15 GB in a gzip compressed version. The aim of this thesis is to further unravel the structure of the pggb generated HPRC pangenome and to develop a tool that can extract genes from all samples. For this purpose, a separate mini GFA file is written for each gene.

2.3 vg

As previously stated, a linear reference genome is limited in representing genetic diversity. To circumvent the problem, so-called variation graphs are used, which is a sequence graph with an additional set of paths representing possible traversals (see Section 2.2). Prior genome graph software included scalable and topological restrictions. In 2014, the tool vg was published, which is a computer-based toolkit that enables the creation, manipulation, and use of variation graphs as references for mapping DNA sequences. One of the biggest applications is to map reads to a variation graph, which however is not used in the context of this work. However, it is recommended to use vg if you want to map your sequenced genome data to the gene specific Mini GFA files generated by the provided pan2 workflow. In this thesis, vg is used in this thesis to extract the variants in VCF format from the Mini GFA files. In addition, the haplotypes of all included sample pathways are determined using vg. For this purpose, all sample sequences are output in Fasta format and then checked for identical sequences to be combined into haplotypes. Version 1.47.0 of vg published in March 2023 is used in this thesis. [8, 16]

3 Methods

The goal of the workflow is to extract the gene sequences of all samples as well as their variations from the pangenome GFA. Additionally, the output contains a multiple-sequence-alignment of all contained haplotypes. This project is solved with three separate python tools: pan2prep, pan2index and pan2gene (see Figure 3.1). After traversing through the pangenome GFA, a so-called Gene Index file is created, which contains the subpaths of all samples for all genes. Then, a Mini GFA file is created for each gene within the Gene Index file, which contains all necessary segments, paths and links. Finally, for each gene, a VCF file containing the variants and a multiple-sequence alignment is created.

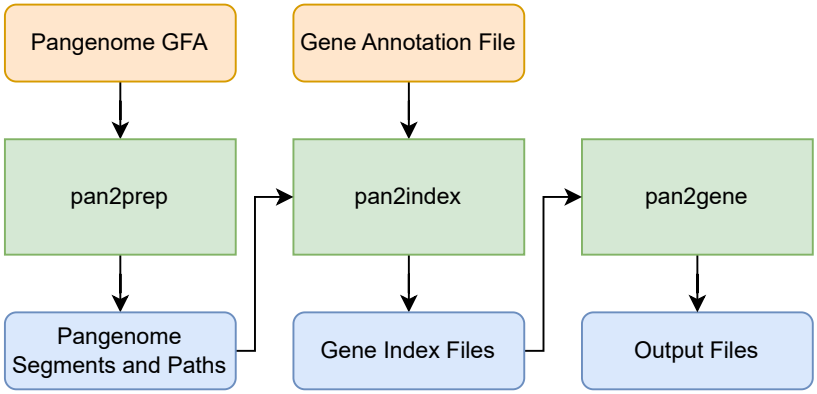

Figure 3.1: **Whole pan2 Workflow.** pan2prep returns the segment and path data extracted from the pangenome. With this pan2index produces the Gene Index files with which pan2index then generates the output files.

In pan2prep (see Section 3.1), all gene independent data is extracted to speed up the runtime of the other two tools. In pan2index (see Section 3.2), the previously mentioned Gene Index files are created for each gene. With these, pan2gene (see Section 3.3) creates the final output files (Mini GFA, VCF, etc).

A path for which annotated gene data is available is used as the reference path. The path used is GRCh38. Note that the CHM13 path can also be used. In this case, the GRCh38 path is preferred as the reference path because it is represented in all chromosomes. As explained in Section 2.1.2, there is no Y chromosome in CHM13.

There are alternative methods for this task, but they have certain disadvantages for the application in this case. For example, a subgraph can be extracted from a GFA file using the vg Toolkit's chunk command. This would be equivalent to the Mini GFA file creation for one gene. However, this subgraph will then contain all paths that contain at least one segment of the gene and all segments in these graphs. In the end, almost

the entire chromosome would be extracted from the pangenome GFA for each gene. The extraction of mini GFA files for genes from a large pangenome GFA file is a gap in previous GFA tools that is closed with the pan2 workflow.

3.1 pan2prep

The task of pan2prep is to extract all gene-independent data from the pangenome. In order to use the data in the pangenome, it must first be read in and put into an easily accessible structure. Then a data structure needed for pan2index is created by preprocessing the reference path. Since it is gene independent, it is created in pan2prep. This will speed up the runtime in case you call pan2index more often for different genes. For the call of pan2prep only one parameter is needed, namely the path to the pangenome file. This is entered after the flag -p or --gfa_file. In Figure 3.2 the general workflow of pan2prep is shown.

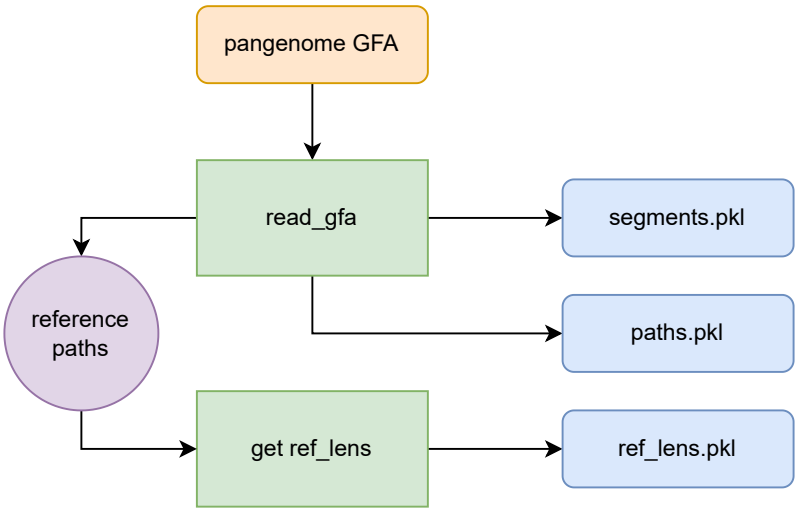

Figure 3.2: **pan2prep Workflow.** Orange represents the inputs and blue the outputs. The purple circle is an internal state.

3.1.1 Data Import

The first step is to import the pangenome GFA file. Since any information in the file could be required, it must be possible to access any information directly. Thus, all segments and paths must be stored. Dictionaries / hashmaps are used for this purpose. It requires longer runtime to insert elements into the data structure, but all data can be accessed in a constant time. Saving the contained links is not necessary as further discussed in Section 3.3.1. Since the entire about 95 GB large file would overload the memory of most local computers and also result in poorer overall runtime, the data is stored more memory efficiently. For all segments only the start and end byte of their sequence of their line in the pangenome file is stored and for all paths only the start and end byte of their segment list. The necessary data can then be read directly from the pangenome and are thus only loaded in the memory for a short time. The following data structures are created:

- `segments` uses the segment id as key and contains the byte position, defined as start and end byte, of the sequence of the segment as value. A separate dictionary for each chromosome is created.
- `paths` stores a dictionary for each chromosome with the path id as key and the byte position in the pangenome as value.

In order to perform the cost effective creation of the dictionaries only once, `segments` and `paths` are written as a binary file with the module `pickle` at the initial call. This can serialize and de-serialize Python object structures. This way the dictionary can be read in immediately as this and no complex hash functions have to be performed. This will be useful if more genes are to be extracted from the same pangenome. Overall, when traversing through the file, only a constant number of operations are performed for each line, resulting in a linear runtime.

3.1.2 Reference Path Preprocessing

In addition, two more gene-independent structures are created, which are later used in subsection 3.2.1 to identify genes on the reference path (GRCh38 path). Annotations for genes in the GRCh38 reference genome use start and end position expressed by the number of nucleotides up to this position. However, the reference path in the pangenome contains only a list of segment ids. So the nucleotide position must be converted to the segment id position.

To do this, first the GRCh38 path for each chromosome must be extracted. This is easily done as the start and end byte is contained in the previously created `paths` dictionary. The next step is to iterate over each segment in the GRCh38 path and get its start and end byte from the segments dictionary. This allows the length of the segment to be calculated by subtracting the end byte from the start byte without actually having to read out the sequence again. One byte stores exactly one character of the sequence.

Thus, it is iterated over each segment in the reference path. The lengths of the segments are added up cumulatively and stored in the dictionary `ref_lens`. This assigns the corresponding segment and its index in the reference path to the cumulative length of the path until the start of a segment.

Since the pangenome is divided into 25 chromosomes, there are also 25 reference GRCh38 paths and thus also 25 entries of `ref_lens`. Now every gene independent data of the pangenome can be accessed.

3.2 pan2index

Once the data has been read in via the pangenome file itself or via previously created index files, the next step is to create the gene indexes. Of the three tools, pan2index is the most complex. The goal is to extract a gene sequence consisting of segment ids from the reference path and to find a counterpart in the other paths, the sample paths. In Figure 3.3 the workflow is displayed.

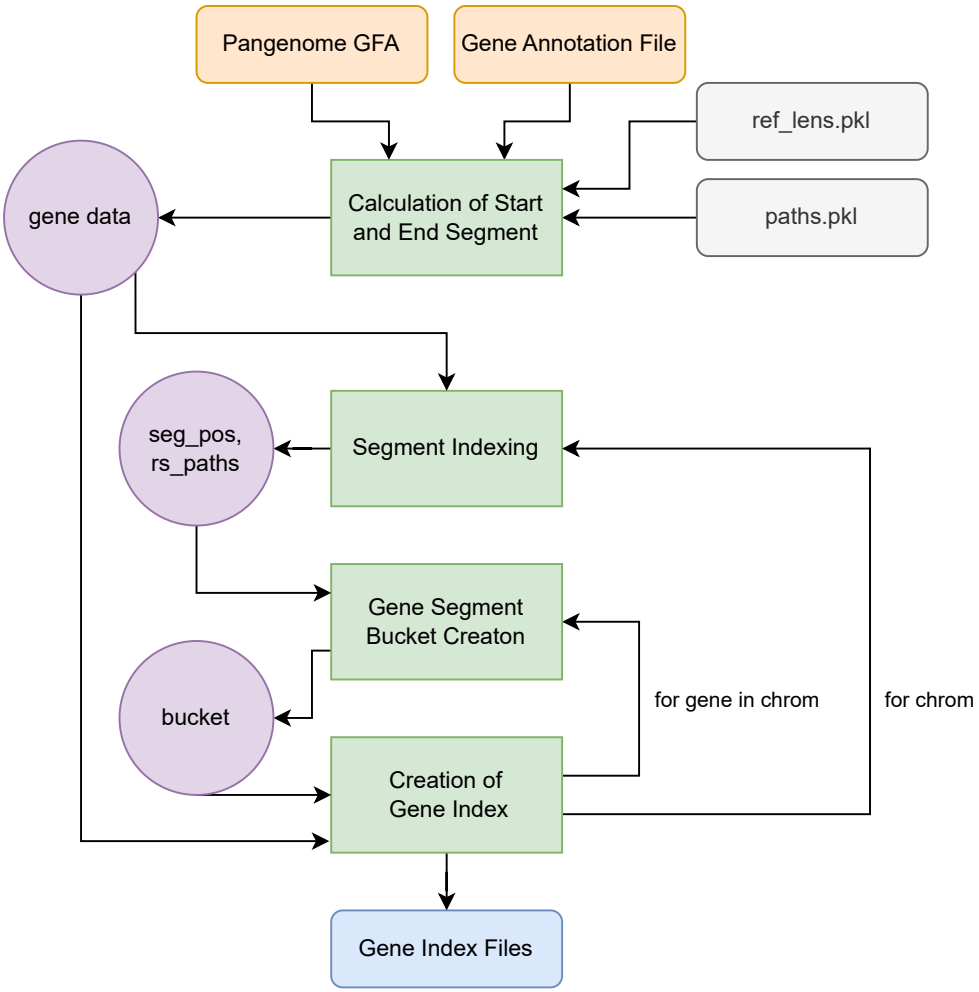

Figure 3.3: **pan2index Workflow.** Only the pangenome GFA and a gene annotation file is necessary to enter manually as input. The white files were previously created by `pan2prep` and will be imported automatically. For each gene a gene index file is written at the end.

There are two ways to import the genes. Either by an GTF annotation file if you are interested in many genes or a self-created file if you are only interested in a few selected genes. If you choose the annotation option, you can use the -j flag to restrict the selection to protein coding genes. In addition, the -n option can be used to specify a comma-separated list of chromosomes of interest. If none is specified, all chromosomes in the annotation file will be processed. To reference specific chromosomes with -n, use the numbers from 1 to 22, as well as X, Y, and M. If a self-created file is used as gene input, it must have the following format: Each line represents a gene and contains tab-separated in this order gene id, start, end, strand and the chromosome. For the chromosome use the same representation as for -n. Use 1 for the positive strand and -1 for the negative. Note that all output files created by pan2index and pan2gene in Section 3.3 will be named after the given gene name. In case of the -a annotation file input, the file name will be a concatenation of the gene name and its ensembl gene id: In this case it will look like this: ATF3_ENSG00000162772.17.<file suffix>. All necessary and possible inputs of pan2index are shown in Table 3.1.

Parameter	Required	Default	Description
-p / --gfa_path	T	-	path to the pangenome GFA file
-g / --genes_path	T/F	-	path to self-created gene annotation
-a / --annotation_path	T/F	-	path to VCF annotation file
-j / --just_protein_coding	F	False	set flag if you are only interested in protein coding genes
-n / --chromosomes	F	all	list of chromosomes of interest
-r / --promoter_length	F	0	length of included region in front of each gene
-m / --max_val_size	F	10000	max value of intervals used in Section 3.2.4
-t / --n_processes	F	1	number of used multiprocesses

Table 3.1: **pan2index Inputs.** T/F means that one of those arguments is required. The -j option has only an effect in combination with the -a flag.

3.2.1 Calculation of Start and End Segment

The first step in creating the Gene Index file is to identify the start and end segments of each gene in the reference path. For this, the dictionary `ref_lens` is used, which contains the starting positions of all segments as well as their segment ids and their index in the reference path (see Section 3.1.2). For the given gene start position `start` the next smaller start position of a segment needs to be found. And then with this as key, the corresponding segment can be found using `ref_lens`.
This is done using the `bisect` module which returns, for a given element and sorted list, the position where the element would have to be inserted in order for the list to remain sorted. A dictionary cannot be referenced via an index, but the order of the inserted elements is still preserved. That means the list of all keys of `ref_lens` corresponds to a list sorted in ascending order. This list will be called `sorted_lens` from now on. The index of the start position of the start element is then the index calculated by `bisect` -1 using `sorted_lens`. Using `ref_lens`, this index can now be used to determine the start segment id as well as its position in the reference path. This can be seen in Figure 3.4. The calculation of the end segment is done in the same way. If an input parameter -r / --promoter_length has been set, the region before the actual gene is also identified as the gene. In this case, the information about the promoter sequence of the gene is also processed. For genes on the positive strand, the value of `start` is reduced by the given promoter_length and for genes on the reverse strand, `end` is increased.

It is important to note, that in most cases a gene does not start at the first position of the start segment, but within the segment. This means the start position of the gene in the start segment, in the following called `start_in_gene`, must also be calculated. Accordingly, we can calculate it easily as follows: `start_in_gene` = $start - sorted\_lens\_entry$. In Figure 3.4 this would be $131142 - 131128$ resulting in a value for `start_in_gene` of 14. Again, the calculation of `end_in_gene` for the end segment works analogously. Note that if `end_in_gene` $= 0$, the end sequence will have length zero. This would cause later the creation of the Mini GFA file (**see Section 3.3.1**) to contain a segment with no sequence, resulting in an invalidity in the format. Therefore in this case the segment with the next smaller index in `ref_lens` is used as end segment and `end_in_gene` is set to the length of the segment. For each gene, start and end segment, their position in the reference path and `start_in_gene` as well as `end_in_gene` are now stored. The term segment sequence is used in the following to describe a sequence of segment ids that also contains a direction. So the same structure as a path has: 104-,105-,106-. This does not refer to the nucleotide

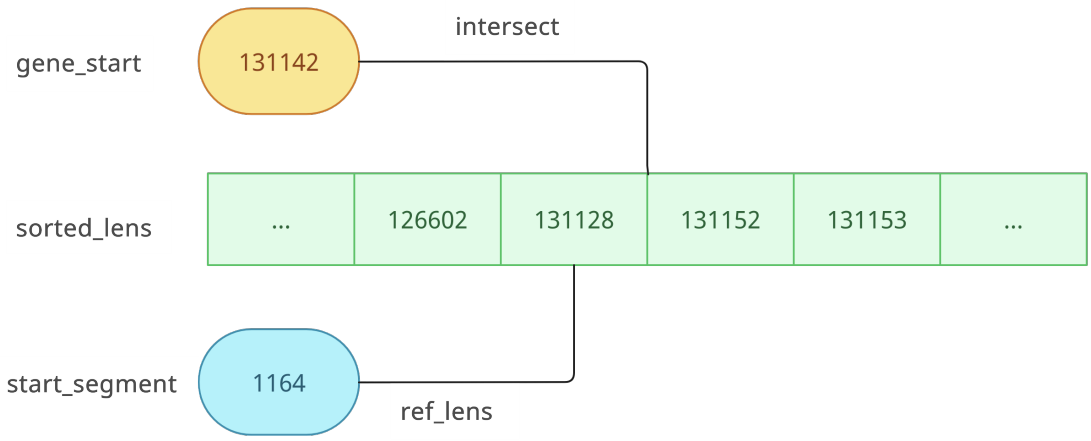

Figure 3.4: **Get start segment id of reference path.** The given value for the start position of the gene is converted to the id of the start segment using bisect, sorted_lens and ref_lens.

sequence that is assigned to each segment id.

3.2.2 Segment Indexing

The reference segment sequence of a gene was determined in Section 3.2.1 by the positions of the start and end segments. Now the next step is to identify the gene in the other sample paths. As defined in Section 2.2, a sample describes in our use case one of the 88 haplotypes of the 44 individuals. A sample path is therefore any path that is not the reference path. Since the CHM13 path is not used as a reference path, it is also treated as a sample path. The goal is to obtain a segment sequence that represents the gene for each sample. A naive approach would be to go through all paths for each gene and extract the most promising segment sequence for each sample. To optimize this approach, only subpaths that meet certain requirements should be searched for the gene. This is done by Segment Indexing.

The central idea is to store for each segment in the reference gene sequence each occurrence in a sample path and then consider only the regions with high accumulations. Thus, a dictionary `seg_pos` is created with a segment id as the key and a list of paths and positions in that path as the value. For this, an entry is created in `seg_pos` for each segment of interest. Then iterate over all paths and whenever a segment is contained in `seg_pos`, an entry is added with the path id and the position of the segment in this path. Note that the check to see if a segment is contained in a dictionary is in O(1). The iteration over the reference path can be skipped. The sample paths can be from the same strand as the reference path as well as from the opposite strand. That means, all matches with the same segment id are added, regardless of their direction (remember: a segment sequence of a path looks like this: 141-,143-,144-,...).

The question remains, which segments are considered to be segments of interest and thus get an entry in `seg_pos`. For example, you could take all segments in the reference path. This would have the advantage that it would be gene-independent and could therefore be calculated in pan2prep. For resource-saving reasons, however, the segments

of interest will be further restricted: Only segments contained in the reference subpath of a gene are used. We name the segment sequence of a gene extracted from the reference path `rs_path` for ref-sub-path hereafter. A `rs_path` can be easily extracted using the start and end positions of the respective gene in the reference path determined in Section 3.2.1. Since all `rs_paths` are necessary in the further course anyway, we save them now and use the contained segments as segments of interest for Segment Indexing.

Since the segment sets of the chromosomes are disjoint and the gene is only expected in the same chromosome as its located reference path, Segment Indexing is performed separately for each chromosome.

3.2.3 Gene Segment Bucket Creation

For a chromosome, the dictionary `seg_pos` has now been created. The steps performed separately in this subsection and the next Section 3.2.4 are performed for each gene. For this purpose, a new dictionary with samples as key `bucket` is created for each gene. The dictionary `seg_pos` contains for each segment in a `rs_path` each position in each path where it occurs. The `bucket` should now be filled so that for each sample we have a list of all positions in the path in which a segment of the gene-specific `rs_path` occurs. As explained in Section 2.2, a sample may include multiple paths representing contigs. In the end, of course, we want to have the best match for a gene for each sample, not for each contig of each sample. Therefore, `bucket` actually stores all contained contig paths for each sample and for these the positions where a segment of `rs_path` occurs.

This is done by iterating over each segment occurring in `rs_path` and retrieving the information contained in `seg_pos`. This is then converted and inserted into the `bucket` structure described earlier. To speed up the runtime a bit, segments that occur too often in all paths are skipped. Since such a segment is so common, it is not suitable for identifying gene locations. A cutoff from 1000 occurrences has proven to be a quite good value for this.

Here is a short example to better understand the structures of `seg_pos` and `bucket`. In `seg_pos` an entry is created for each segment contained in a `rs_path` of all genes. For each segment, the position of the segment is saved for each path. `bucket` is calculated for each gene individually. Here, for each path, the positions of all segments that are in the `rs_path` of the gene are stored. In the following example, seg1, seg2 and seg3 stand for segment ids and p1, p2 and p3 for path ids. The `rs_path` of the gene for which the `bucket` is created contains only the segments seg1 and seg3. All three paths are contigs of the sample sample1. Note that it is not uncommon for a GFA graph to be cyclic, that is, to contain a segment more than once in a path.

$$\text{seg\_pos} = \begin{cases} \quad \dots \\ seg1 : \{p1 : [5, 75, 113], p2 : [20, 110], p3 : [1751]\} \\ seg2 : \{p1 : [100], p3 : [15, 75]\} \\ seg3 : \{p1 : [6, 76], p2 : [22, 111], p3 : [1749]\} \\ \quad \dots \end{cases}$$

$$\text{bucket}[sample1] = \begin{cases} p1 : [5, 75, 113, 6, 76] \\ p2 : [20, 110, 22, 111] \\ p3 : [1751, 1749] \end{cases}$$

3.2.4 Subpath Extraction from Non-Ref-Paths

The next part of pan2index can be challenging to keep track of. Therefore, a pseudo code is provided in Figure A.13 in the appendix. For each gene, a `bucket` was extracted from `seg_pos`. This stores information about possible occurrence of a gene in the non-reference paths. The further approach is to read out the most promising region from `bucket` and then extract the actual subpath for each sample from it. The following steps are performed for each sample in the `bucket`.

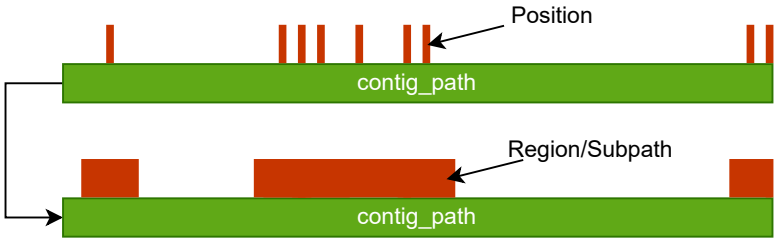

Figure 3.5: **Merging of Intervals around Bucket Positions.** Intervals are spanned around the positions in bucket. Overlapping intervals are merged into regions, which are then scorched for the probability of containing the gene.

For a path, `bucket` simply stores a list of index positions in the path. The task is to combine adjacent positions to generate subpaths (displayed in Figure 3.5). A subpath can potentially contain the searched gene. For each index position of a path we create an interval with an integer `span`. Let i be an index position of a path, then [i - `span`, i + `span`] is the corresponding interval. Now all overlapping intervals are merged into one large interval. For this purpose the library `portion` is used, which is designed for operations involving intervals. We call the result intervals created from the initial intervals by merging them `merged_intervals`. Due to `span`, intervals can also extend beyond the border of a path. In this case, the borders of the interval would be trimmed so that each interval is contained completely within the path. Here is a short example with `span` = 5 for a path of length 1002 in bucket. The length of a path is defined as the number of contained segments.

$$\texttt{bucket[sample][contig]} = [16, 17, 19, 22, 1000]$$
$$\text{intervals with } \texttt{span} = [[11, 21], [12, 22], [14, 27], [995, 1005]]$$
$$\texttt{merged\_intervals} = [[11, 27], [995, 1001]]$$

So in this example, we have two `merged_intervals` that are possible regions containing the gene. The second interval has extended over the borders of the path and was therefore shortened so that the end of the interval is also the last position in the path. Since many positions stored in `bucket` are very close together such as 16,17,19, many intervals can be ignored for merging because there is enough overlap. The -m input parameter can be used to specify a maximum interval size which limits the number of intervals. If there is a large number of intervals, the calculation of `merged_intervals` may take more time. Therefore, intervals are randomly sorted out until the maximum interval size is reached. This case is only relevant for very large genes, which is why the default is set to 10,000.

Next, a method was developed to compute a `score` for each `merged_interval`. The higher the `score`, the more likely the region found contains the gene being searched for. For each sample, the region with the highest `score` is saved and overwritten if a new region has a higher `score`. This procedure is not limited to the `merged_intervals` of one path, but to all contigs of a sample. There are several approaches how to determine a `score` for a region. The most accurate approach would be to perform a sequence alignment between the subpath of the region and the `rs_path`. Furthermore, for a region, a sequence alignment to the reverse complement of the `rs_path` would also have to be performed, since the contig could also be a read of the reverse strand. The reverse complement of a path describes the reversal of the order and direction of the segments. So the reverse complement to 20+,21+,22+,26+ is 26-,22-,21-,20-. However, later in Figure 4.14 it becomes clear that a `rs_path` of a gene can span more than 10^4 segments. An exact sequence alignment algorithm fills a matrix of size m * n, where m and n are the lengths of the two segment sequences. This would result in quadratic runtime for each entry in `merged_intervals` for each path and each gene. Therefore, a less runtime intensive method is used to determine the `score`, which, however, is only an approximation of the score from the sequence alignment: A modified form of the Jaccard Index. It is used to calculate similarity between two sets and is the length of the intersection divided by the length of the union. It returns a value between 0 and 1. Here, instead of the union size of the sets, the length of the interval is used. The size of the intersection of the sets is the number of intervals that have been merged into one entry in `merged_intervals`, since an interval represents a segment contained in `rs_path`. So when creating `merged_intervals`, for each merged interval it is counted how many initial intervals were merged. In the end, only the number of segment matches is scored, but not their order. So, in the above example, for the interval [11, 27] we would have a `score` of $3/(27-11) = 0.1875$ and for the interval [955,1005] we would have a `score` of $1/(1001-995) = 0.1\overline{6}$. So here the first interval would be stored as the best match, since it has a higher `score` than the other; assuming there was not already an even better `score` in a previous contig of the sample. The difference was only so small in this example because the second interval was previously shortened due

to the exceeding of the path borders. Most genes contain much more segments, which means that the important `merged_intervals` are also much larger. Therefore, in most cases, the effect of interval shortening is not as significant as it is here.

If all contigs of a sample contained in `bucket` have been traversed and a region with the highest `score` has been identified, the actual segment sequence is now extracted from this region. Since the creation of `seg_pos` in Section 3.2.2, abstract index positions and frequencies have been used. Now an actual sequence of segment ids is required. So with the `paths` dictionary, the path of the contig containing the region with the highest `score` is read from the pangenome GFA file. Reminder: The index boundaries of an interval represent the corresponding positions in the path. From this the subpath is extracted with the start and end position of the best scoring interval. This is called `ss_path` for sample-sub-path.

To find the start and end of the gene in the `ss_path` is a challenging procedure, since no segment of `rs_path` can be assumed to be safely contained in the `ss_path`. In addition, there may be deletions of subpaths in `rs_path` or insertions in `ss_path` that are longer than `span`. In this case, the gene would be only partially contained in the `ss_path`. Furthermore, the gene may also be split between two contigs. By using the fast but less accurate scoring method of the Jaccard Index, a region can also be incorrectly identified as a gene if it contains more segments of the `rs_path` than the actual gene region. This is due to the fact that the order of the segments is not scored. Thus, several steps are applied to optimize the chance of correctly identifying the start and end of the gene. For this purpose, we first assume that a gene is completely contained in the `ss_path`. In order to detect the start or the end of the gene, we compare the path with the start and end segment of the `rs_path`, which contains the correct segment sequence of the gene from the GRCh38 path. Since there may be a genetic variant in one of these segments in the `ss_path`, the first and last three segments of each `rs_path` are stored for comparison, if the gene contains that many segments, otherwise only the first and last segments, respectively, are used. These segments will be referenced as `start_segs` and `end_segs` from now on. So, both a segment in `start_segs` and a segment in `end_segs` must be found in `ss_path`. Then the segment sequence in between can be used as the detected gene.

It is important to identify the strand of the contig and thus of the `ss_path`. Only then can the start and end segments be searched for correctly. For this purpose two sets are created: One that contains all elements of the `rs_path` with their direction, and one that obtains all elements with their reverse direction. These two sets are independent of the `ss_path`. Therefore, they are already created beforehand directly after the creation of the `bucket`. Now, each segment in the `ss_path` is tested to see if it is included in one of the two sets. The search in a set runs in O(1). The number of segments contained in a set is also counted. The set with more matches decides whether the contig is on the reverse strand or not. In most cases, however, one of the two counters is 0. If the `ss_path` is on the reverse strand, this means that first the end segment of the `rs_path` is expected in reverse direction and only then the start segment. So in this case we swap `start_segs` with `end_segs` and also form the reverse complement of them.

To find a start segment, it is iterated over the segments of the `ss_path` from the left. If the first element in `start_segs` is found, the position is saved and the iteration is terminated. If the second or third element is found without the first one being found before, the iteration is also stopped and the current position -1 or -2 is returned. In

this case it is assumed that the segment before is a mutation of the first start segment. If the `ss_path` was not cropped for exceeding the path borders, a match with a start segment is expected approximately at position = `span`. In case the gene contains the start segment more than once, the correct one (i.e. the first one) would still be found, since the path is traversed from the left. However, if the start segment also happens to appear directly before the gene, this would be identified incorrectly as the start of the gene. When iteration continues to a certain cutoff value without being stopped by finding a start segment, this means that the start segment is probably not contained in `ss_path`. This implies that one of these cases is present:

- Case 1: There is a too large insertion at the position in the contig path or a too large deletion in `rs_path`, so that `ss_path` does not contain the beginning of the gene.
- Case 2: `ss_path` contains only the second part of the gene because the first part is on a different contig.
- Case 3: The gene has a mutation in the start segments, which means that they are not included. In this case, identification of the start of a gene is not possible.

As cutoff 2 * `span` is used. To validate case 1 and 2 the contig path is now traversed backwards from the start position of `ss_path` to search for a start segment. If this is found, the position is returned as the new start position of the gene. If the border of the contig is reached, it is checked whether the last segments before the border are contained in the `rs_path`. If so, it is assumed that the gene is distributed on several contigs. In this case a flag '*' is set. If not, pan2index cannot give exact information about the start of the gene, the original start position of `ss_path` is used and the flag '?' is set. The search for the end segment proceeds in the same way, except that `ss_path` is iterated backwards and if no end is found, the rest of the contig path is iterated forwards (see Figure 3.6).

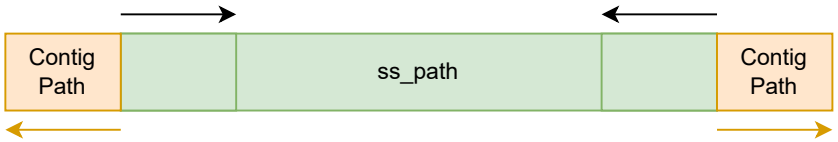

Figure 3.6: **Search of start and end segment in the ss_path.** The green section represents the ss_path. First this is traversed according to the black arrows until a searched segment is found or the cutoff border is reached. If no segment is found, the orange contig path is searched for the searched start/end segments in the specified direction.

Now the start and end position of the gene are determined for each sample in `bucket`. This is now written to the Gene Index File. The file is structured as follows: first comes a header line, which gives information about the strand and the chromosome of the gene. In addition, the `start` and `end_in_gene` values calculated in Section 3.2.1 and the start and end segment of the `rs_path` are also written to the file. They will be needed later when creating the Mini GFA files. After that, each line contains a sample id and the segment sequence extracted for it. If a '*' or '?' flag is set, it will be

written after the sample id. The flags give the user the flexibility to filter out these lines easily dependent on their use case. The entry for the reference path is in the last line. Note that all this was possible without using the actual nucleotide sequences. Only the segment ids were necessary. Since some segment sequences were generated from a reverse strand contig, the file would now contain some reverse complement paths. This complicates the use of the generated data, therefore all paths are rotated so that they point in the same direction. If the gene is located on the opposite strand, the paths should also be reversed. Each generated path of a gene is rotated according to the pattern in Table 3.2.

	contig positive	contig negative
gene strand = +	-	reverse
gene strand = -	reverse	-

Table 3.2: **Approach whether the reverse complement of a path should be written into the file.** In the last case the path would be reversed twice, which would result in the original path again.

The choice of the value of `span` is based on these criteria: If `span` is too small, a `ss_path` contains only part of the gene if the deletions or insertions are too large. If `span` is too large, an unnecessarily long time is spent on irrelevant subregions. Here `span` = 500 is used. A more extensive evaluation of the optimal value for `span` might improve the algorithm performance further, but is out of the scope of this thesis.

Now a Gene Index File has been created for a gene that can be further used by pan2gene. The generated paths for each sample will later be the paths in the Mini GFA file that is created for each gene. In the pan2index workflow, the next gene is processed, which starts again with the creation of the `bucket` in Section 3.2.3. Apart from creating the `bucket` and reading a contig path from the pangenome GFA file, each gene can be processed in parallel for the procedure in this subsection. In the current version of Python, only one thread can work in parallel at a time because of the GIL (Global Interpreter Lock). Therefore the multiprocess module is used which bypasses the GIL. With the parameter -t the desired number of parallel active multiprocesses can be specified.

3.3 pan2gene

Before, both the gene-independent general pangenome data and the gene-specific path data were extracted from the pangenome. It is the task of the third and final tool, pan2gene, to output this data in useful formats. For each gene, a GFA file (Mini GFA) with the graph structure, a VCF file with the variants, a Fasta file with the haplotype sequences and a MSA file with a multiple sequence alignment will be created. The workflow of pan2gene is visualized in Figure 3.7. In Table 3.3 the parameters for a call of pan2gene are described. All gene input parameters (-g/-a/-j/-n) are optional. By default, all available Gene Index files are processed. The gene input option applies in case the pan2index into pan2gene workflow is run for the second time with different genes. In this case, only output files for the mentioned genes will be created and not for those that have already been processed.

Parameter	Required	Default	Description
-p / --gfa_path	T	-	path to the pangenome GFA file
-v / --vg_path	T	-	path to your installed vg executable
-g / --genes_path	F	-	path to self-created gene annotation
-a / --annotation_path	F	-	path to VCF annotation file
-j / --just_protein_coding	F	False	set flag if you are only interested in protein coding genes
-n / --chromosomes	F	all	list of chromosomes of interest

Table 3.3: **pan2gene Inputs.** The path to the pangenome GFA and the path ot the vg executable are the only required options .The -j option has only an effect in combination with the -a flag.

3.3.1 Mini GFA Creation

Now that the Gene Indexes have been created, it is easy to create the first output files: the Mini GFA files. As explained in Section 2.2, a GFA file contains segments, paths, links and a header. All the information needed for the paths is provided directly in the Gene Index File. All segments contained in the paths are written to the file with their sequences using the `segments` dictionary. The necessary links can be found easily as well: While iterating over the segments of all paths, a link is created for two consecutive segments. Duplicates are eliminated. In an alternative approach, one could also output the links in pan2prep in an additional dictionary and then read them back in here. Then all links between segments contained in the Mini GFA could be used. In addition to longer runtime and more memory, this would also have this disadvantage: If a pair of segments occurs again in another path, there would potentially be more links than the Mini GFA Variations Graph should contain.

Since exclusively the gene sequences should be contained and most genes start or end within a segment, the sequences of the start and end segment must be cut at the right position. For this purpose, the `start_in_gene` and `end_in_gene` values created in Section 3.2.1 are used. In the start segment the partial sequence until the base with position `start_in_gene` has to be removed.

$$seq = seq[start\_in\_gene :]$$

With the End segment, it's the other way around: the suffix of the sequence must be cut off:

$$seq = seq[: end\_in\_gene]$$

In python, ":" in this case means "from here to the end" if it is at the end, or "from start to here" if it is at the beginning. If the gene is on the reverse strand, the identified start segment is at the end of the path and the end segment is at the beginning. In this case, the front part of the end segment and the back part of the start segment are cut off. To do this, start and end segments and `start_in_gene` and `end_in_gene` are swapped. Furthermore, if the gene is on the opposite strand, `start_in_gene` is changed to `segment_length - start_in_gene` to cut at the right place. It works the same way for the end segment. For a better overview of the outputs, an output folder is created for each chromosome. A Mini GFA file is named after the gene id and added to the corresponding output folder.

After the Mini GFA file creation, more output files are created for each gene: A VCF

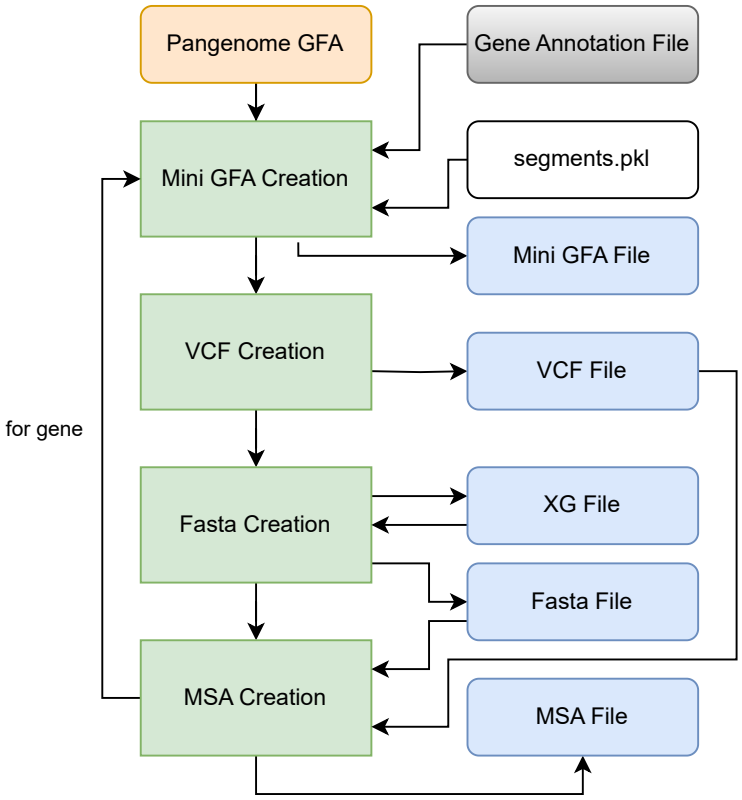

Figure 3.7: **pan2gene workflow.** Orange describes the manually entered input file. The white files are read in automatically. The gray file is an optional input. Output files are displayed in blue. Five of them are created, which are partially reused for further calculation.

file with the variants, a Fasta file with the sequences of all haplotypes and lastly a multiple sequence alignment.

3.3.2 VCF Creation

The Variant Call Format (VCF) contains gene sequence variations. This is one of the key pieces of information stored in the pangenome. The tool vg (Section 2.3) is used to create the VCF file. For the creation of the VCF file the following command is used:

```
./vg deconstruct -t 32 -a -e -P grch38 <GFA file>
```

The -P grch38 flag specifies that the path with the 'grch38' prefix should be used as reference path to which the other paths are compared. Note that the GRCh38 and CHM13 paths are annotated as grch38 and chm13 in the pangenome file. With the flag -a, also variants contained in variants are considered, in case any exist. The flag -e leads to the variants being calculated with the paths instead of the links and -t 32 defines the number of used threads, here 32 threads are used. In addition to the position and the alternative sequences of a variant, the VCF file also contains additional information such as the allele frequency or the subpaths of the traversed segments.

3.3.3 Fasta Creation

The Fasta format is a text-based format that contains the primary structure of multiple DNA or protein sequences. Here the Fasta file should contain the sequences of all included haplotypes. A haplotype describes a combination of genomic variants. In this case, all samples with identical sequences are combined into one haplotype. For this purpose, vg is used as well. Before the sequences can be extracted from the GFA file, vg must convert the GFA file to an XG file. An XG file can be understood as a low-memory version of the graph consisting of indexes. Many vg methods work faster on an XG file than on the GFA file [8]. The corresponding command is as follows:

```
./vg index <GFA file> -x <XG file> -t 32
```

The flag -t 32 means again that the process is started with 32 threads to speed up the conversion. To get the haplotypes, each pathway must first be converted to a sequence. Again, vg has a function:

```
./vg paths -x <XG file> -F
```

The previously created XG file is used as input and the -F flag generates the output in Fasta format. However, this is not written to a file but loaded directly into the RAM. Right now, the gene sequence is included for each sample, not just the haplotypes. To determine the haplotypes of the sample sequences, all sequences are inserted as keys into a dictionary, which receives a count variable as value. This counts how often the key would be inserted to generate allele frequencies. For each haplotype, one of the containing sample ids is used as header for simplicity. For the haplotype that contains the reference path, this is specified. Which other samples have the same haplotype is displayed in a separate table (.haptab file), whose information can then also be used for the multiple sequence alignment. Finally, each haplotype sequence is written to the output Fasta.

The reason why the XG file is not deleted after creating the Fasta is that it can still be used for further vg calls after using pan2gene (for example vg map or vg find).

3.3.4 MSA Creation

A multiple sequence alignment (MSA) is an alignment of three or more biological sequences. However, this problem is NP-complete, which means that it cannot be solved non-deterministically in polynomial time [17]. Thus, creating a new MSA would result in exponential runtime. Fortunately, one of the features of the GFA file can be used to work around this problem: Paths containing the same segments are already aligned at this point. This means that only the variants are not aligned.

The information about the variants can be extracted from the previously created VCF file. There for each variant it is stored, which samples contain it and after which segment it occurs. In addition, the haplotype structure created for the Fasta file is reused. A string is created for each haplotype, which is subsequently expanded block

by block for each segment or variant. To create the MSA, first the GRCh38 path is read from the Mini GFA file. For each segment contained therein, it is inspected whether this is also contained in another haplotype pathway and whether a variant is present here. If it is present in all haplotypes and no variant is present, the sequence associated with the segment is added to all haplotype sequences. If the segment has a negative direction in the path, the reverse complement is used, of course. When a variant is present, a distinction is made between two cases. A variant is represented in VCF format using a reference sequence (ref) and at least one alternative sequence (alt). Case 1 occurs if both the reference sequence and all alternative sequences have the length 1. Then, the corresponding nucleotide is inserted directly in all haplotypes, since an optimal alignment is trivial here. For all haplotypes that do not contain the segments in the variant, a gap symbol '-' is inserted. If one of the sequences in the variant contains more than one nucleotide, case 2 occurs: Here it is not possible to say what the alignment looks like. An MSA tool like clustal omega [18] could be invoked for this case, which does not require a lot of resources for short sequences like [A, AT]. For longer and more sequences, however, the problem of exponential runtime occurs again. For this reason, in our workflow, the areas are not aligned. However, the missing alignment in the MSA is indicated. Therefore, corresponding haplotypes in the previous example would be extended with A-- and -AT. Haplotypes that do not contain the segments at all are extended by n gaps '-', where n is the sum of the lengths of the reference sequence and all alternative sequences. A disadvantage of this approach is that a lot of gaps are used for such a variant [C,CCC,CCCCC,CCCCCCC].

Now all output files for one gene have been created and it is possible to continue with the next gene.

4 Results

In this chapter the results of the analysis of the pangenome GFA file and the output files generated by pan2gene are shown and interpreted.

4.1 Pangenome GFA Analysis

As described in section 2.2, the HPRC pangenome is represented as a GFA file. The structure of this file will be analysed further as follows. A GFA file contains segments, paths and links. Table 4.1 shows the frequencies of the different line types:

line type	**number of lines**	**proportion of lines**	**proportion of file size**
segments	110,884,673	41.74%	10.25%
paths	34,796	0.01%	85.52%
links	154,756,169	58.25%	4.23%

Table 4.1: **Frequencies of the different line types.**

It turns out that although the file contains many more segments and links than paths, the file consists mostly of the paths. The paths must therefore contain a very large number of segments. In total, there are around 10^9 segments. Since a segment id in the HPRC pangenome is named in ascending order of integers, this means that the length of the segment ids is limited to 9 digits and thus 9 bytes. As the length of the segment ids is the only variable factor for the length of a link line, links make up only about 4% of the file size, although they are the most common type. It also makes sense that there are more links than segments, since the variation graph should be densly connected and each variation results in multiple links. Therefore, the difference between the number of segments and the number of links is even smaller than expected. The smallest possible segment line consists of 6 bytes and the smallest possible link line of 12 bytes. Still, the relative file ratio of the segments is higher than that of the links. Since the length of a link line is limited by the maximum segment id length, a link of maximum length requires 27 bytes, while the sequence of a segment can be of unrestricted length. This suggests that there must be segments with sequences significantly longer than 14.

In Figure 4.1 these assumptions are confirmed. There are segments with more than 10^7 bp long sequences. Nevertheless, the median for the length of the segment lines is very low at just over 10 bytes. This means that half of all segments must contain very short sequences. For the links, it is once again confirmed that their length is limited by a constant threshold. Although paths are the longest lines, there are also very short paths. This will be discussed in more detail later. The longest paths reach a size of

over 10^8 bytes, all contained in a single line of the file.

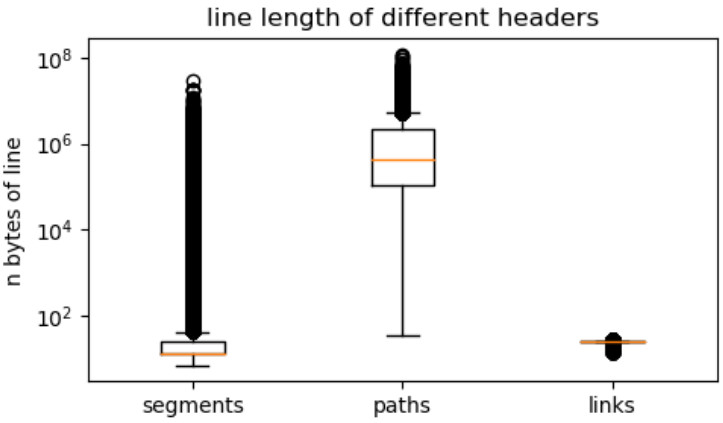

Figure 4.1: **Data Size Comparison of Line Types.**

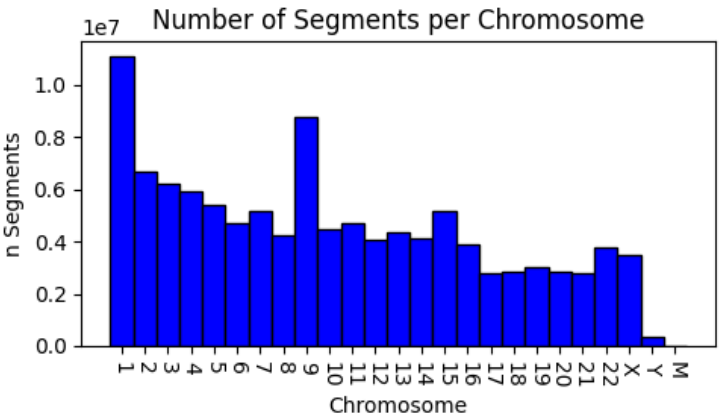

Figure 4.2: **Number of Segments per Chromosome.**

In Figure 4.2 the number of segments per chromosome is displayed. Note that this plot, like most of the following ones, has a logarithmic y-axis. A total of 25 chromosomes are contained in the pangenome file. The 25th chromosome (M) is the mitochondrial chromosome. It contains only about 1000 segments in total, which is the least by far. Chromosome 1 contains the most segments of all chromosomes, with approximately 10^8 segments. With information from Table 4.1, this means chromosome 1 contains about 10% of all segments in the pangenome. Chromosome 1 is also the longest of all chromosomes according to GRCh38 and CHM13. Nevertheless, such a large lead is surprising. Chromosome 2 is only slightly shorter than chromosome 1, but contains significantly less segments. Since the number of segments is calculated from the number of variations, it can be assumed that chromosome 2 contains fewer variations than chromosome 1. It is probably more conserved in terms of evolution. It is also surprising that chromosome 9 contains the second most segments, although it is a rather average-length chromosome. So a lot of mutations seem to have occurred here as well. Chromosome Y contains fewer segments than expected, but this may also be due to the fact that not every sample contains a Y chromosome, as shown in Figure 4.5. In the appendix in Figure A.4, the number of nucleotides per chromosome can in addition be seen. It shows that the number of contained segments correlates with the number of contained nucleotides.

To investigate the segment lengths further, Figure 4.3 and Figure 4.4 show the absolute and relative frequency of the segment lengths. It can be seen that about $7*10^8$ segments contain only one base as sequence. This makes up about 60% of all segments. This means that the majority of all variants are only mutations of a single base. In total, though, there are only four different options for such a segment (A, T, C and G). So a new segment is always used for such a variant. This raises the question of whether segments can occur more than once for different regions and, if so, what the criteria are for this. Furthermore, you can see that the frequency of the segments decreases the higher the corresponding segment lengths are. The value for the segment length 10,000 stands for the sum of the frequency of all segment lengths $\geq$ 10,000. In total, there are about 10^5 segments with such long sequences. The longest is the segment with id 110694287 in chromosome Y and 30,555,432 bp long. It contains a part of the

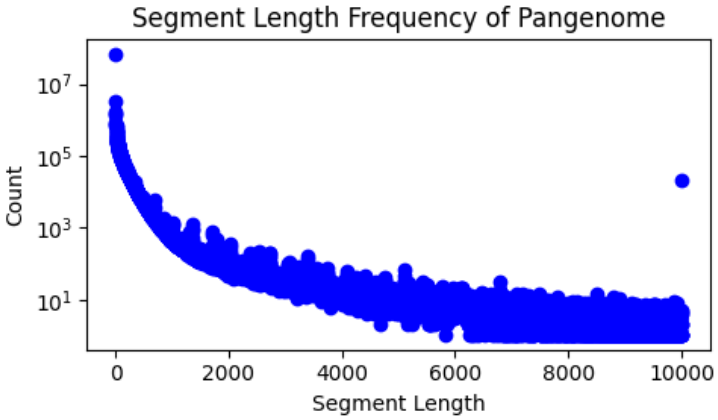

Figure 4.3: **Frequency of Segment Lengths.** All values after 10000 have been merged

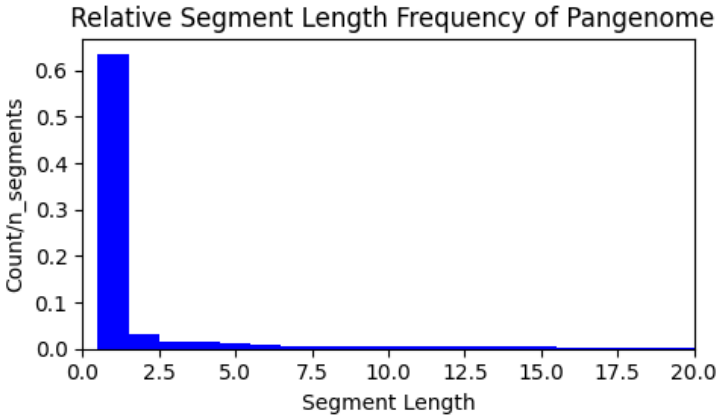

Figure 4.4: **Relative Frequency of Segment Lengths.** Only in the interval [1,20]

telomere of the chromosome and is only contained by the GRCh38 path. Most of the sequence is made up by the gap filler base 'N'. The sequence of this region could not be identified with certainty when the reference genome was created (see Section 2.1.1). An overview of these regions can be found in the appendix in Figure A.1.

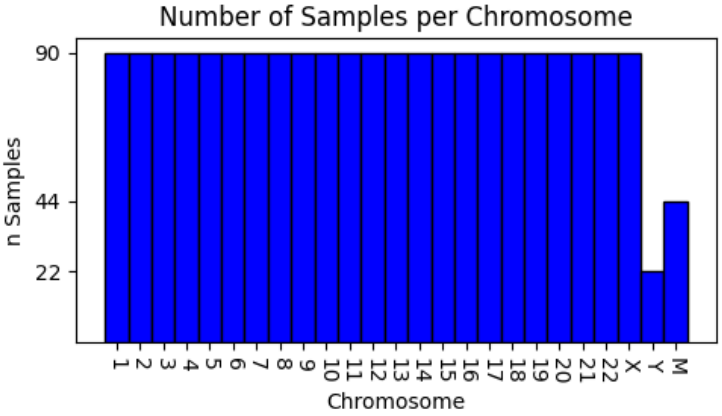

Figure 4.5: **Number of Samples per Chromosome.**

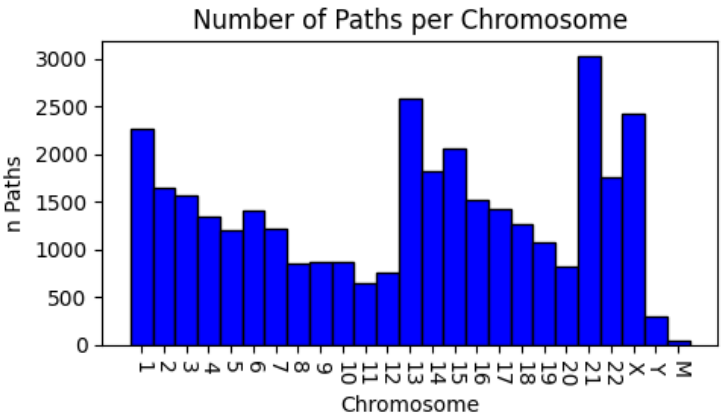

Figure 4.6: **Number of Paths per Chromosome.**

Next, the paths are investigated in more detail. Figure 4.5 shows how many of the samples are present in the chromosomes. All 90 are present in every chromosome except in Y and M. In chromosome Y, as stated in Section 2.1.2, only GRCh38 is included as a reference genome. In the mitochondrial genome, both reference genomes are included. Since this is independent of the paternal and maternal haplotype, it also makes sense that only about half of the samples (with the exception of two) contain chromosome M. Chromosome Y is only contained by 22 samples. This is to be expected if about half of the samples are male, because only one of their haplotypes contains a Y chromosome. The Y chromosome contains relatively few genes in comparison to other chromosomes and is therefore somewhat understudied, especially in large-scale genetic studies [19]. However, it is surprising that all samples contain an X chromosome and

not just about 75%. This will be discussed later in Figure 4.8.

In Figure 4.6 the number of paths per chromosome is shown. Apart from the reference genomes, each sample is divided into several contigs that represent paths. Chromosomes 13, 21 and X have a very high number of contigs. The reads used to create the contigs could probably not be joined together so well. This is the case, for example, with particularly repeat-rich regions. In Figure A.2 it can be seen that only some samples contain an extraordinary amount of contigs in these chromosomes, not all. Apart from that, a chromosome contains on average between 500 and 2500 paths. Chromosome M contains exactly 44 paths. One for each sample.

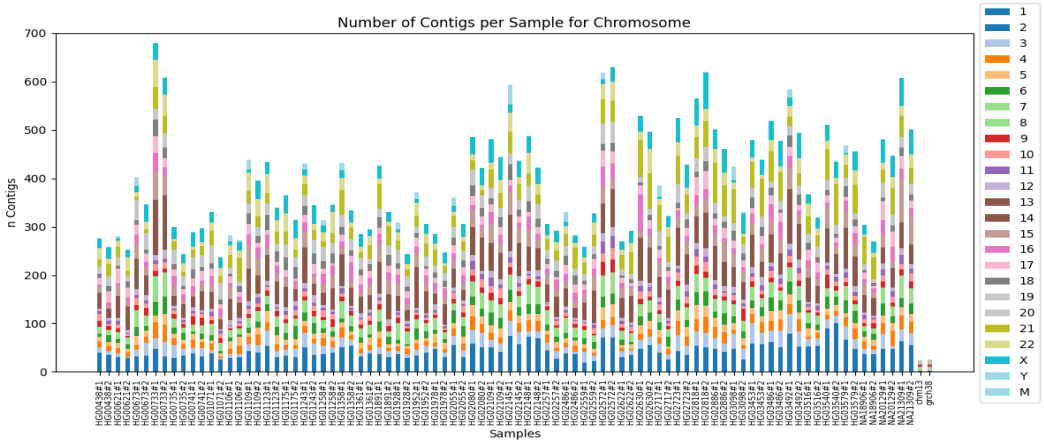

Figure 4.7: **Number of Contigs per Sample.**

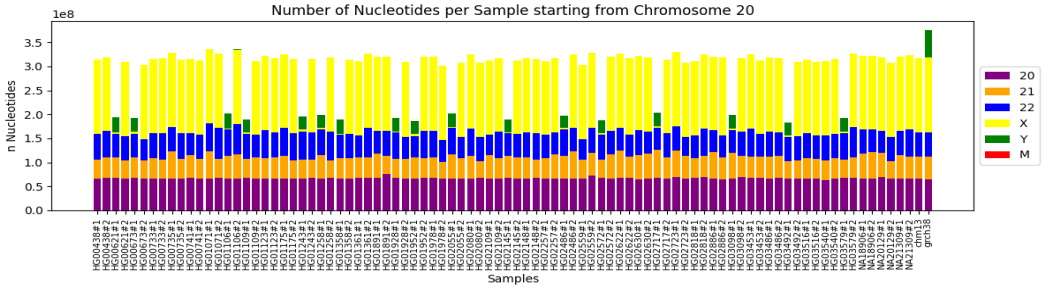

Figure 4.8: **Number of Nucleotides per Sample for Chromosomes 20-22, X, Y, M.**

In Figure 4.7 the number of contigs per sample for all chromosomes is displayed. Some samples contain significantly more contigs than others. Of these samples, no similarity can be established in terms of population: They contain chinese, african, gambian and kinyawan heritage. It remains unclear why their genome ist splittet in more contigs than the other samples. In total, a sample contains around 280 to 680 contigs distributed over all chromosomes. Per definition, the two reference genomes contain only one contig per chromosome. Figure 4.8 shows for each sample the number of bases contained in all its contigs. In addition, only the last chromosomes 20-22, X, Y and M are displayed. It is interesting that all samples containing a Y chromosome have only a very short X chromosome. These sequences map to around the middle of the GRCh38 chromosome X sequence. It is suspected that this small section of the X

chromosome has been incorrectly annotated and actually belongs to chromosome 22 or Y. This explains why all 90 samples contain an X chromosome. It is noted that the Y chromosome of GRCh38 is longer than that of all samples. In the appendix in Figure A.3 it can be seen that the total number of nucleotides contained in each sample is about $3 * 10^9$. The samples with Y chromosome and thus a no X chromosome are slightly shorter.

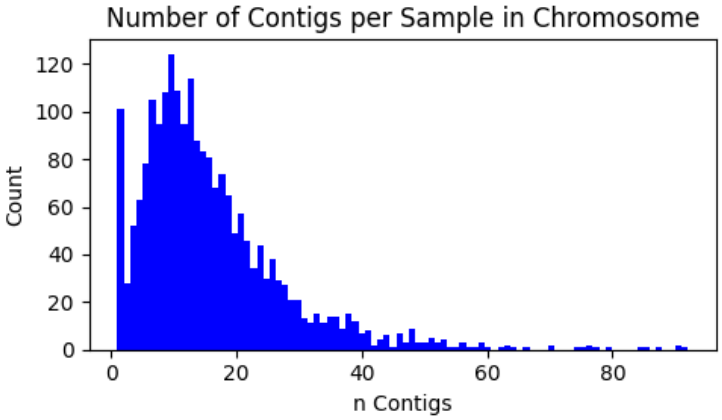

Figure 4.9: **Number of Contigs in Sample per Chromosome.**

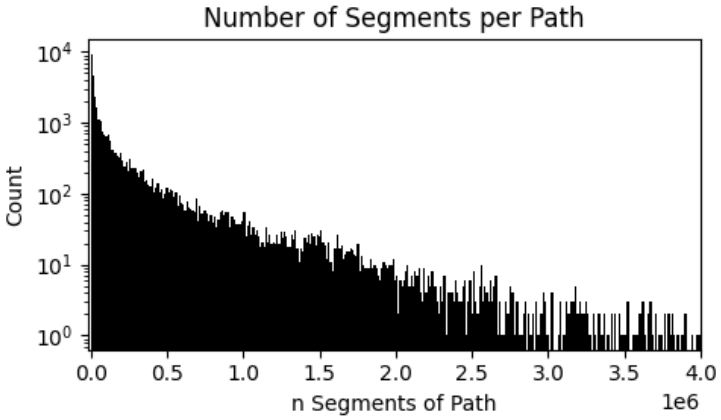

Figure 4.10: **Number of Segments per Path.**

In Figure 4.9 you see the same content as in Figure 4.7, except that it is not separated into samples and chromosomes. Most sampels contain about 5 to 15 contigs per chromosome. 97 times only one contig is contained in a sample in a chromosome. Of these 97 contigs, 49 belong to the reference paths and 42 to the chromosome M paths of the samples, leaving 6 other paths (4 in chr11 and 2 in chr Y). Most samples with more than 50 contigs come from the chromosomes with the most paths 13, 21 and X, as identified in Figure 4.6.

The distribution of the amount of contained segments per path is shown in Figure 4.10. Nearly 10^4 paths contain only one single segment. These segments contain the centomeres of the samples. These are very repeat-rich regions in the middle of a chromosome. Those regions were not included with the rest in the variation graph and do not appear in any other sample or reference path. This means that no sequence from this region can be processed by pan2index. They were probably not included because the sequencing may not be as accurate here. Furthermore, no genes are generally contained in these regions [20]. The first segment in the GRCh38 path in chromosome 1 is 4105. Many of the segments before are such centomeric segments. However, they can also occur afterwards. In the appendix, Figures A. 5-7 visualizes further data on the lengths of the paths.

At last, the links in the pangenome are further explored In Figure 4.11 the number of links per chromosome is displayed. If we compare this with the number of segments per chromosome in Figure 4.2, we notice a strong similarity. The more segments are included, the more links are included as long as the variation graph is connected. The ratio between number of segments per chromosomes and number of links per chromosomes is nearly constant across chromosomes with a factor of around 1.4.. With n segments included, therefore, about 1.4n links are to be expected in the case of 90

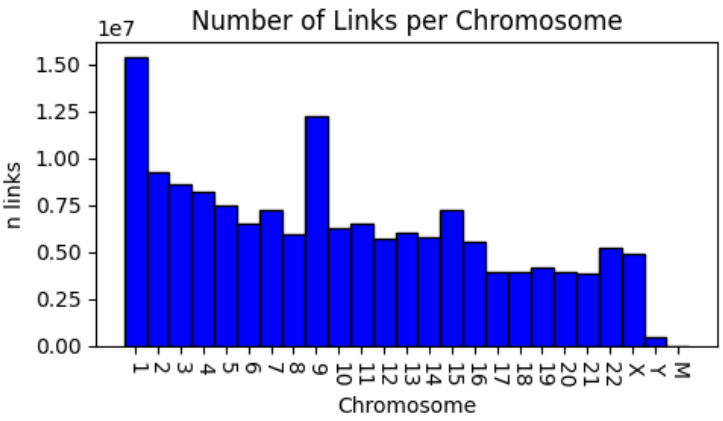

Figure 4.11: **Number of Links per Chromosome.**

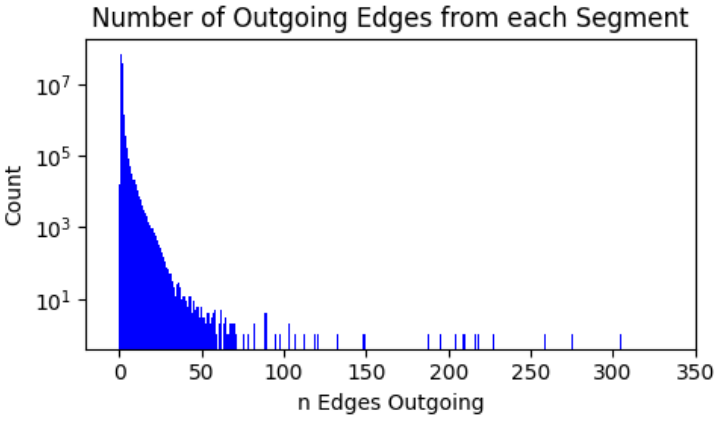

Figure 4.12: **Number of Outgoing Links from each Segment.**

contained samples. Figure 4.12 represents the number of outgoing links per segment. Not a single link originates from about 10^4 segments. The majority of these are the centomereric segments already identified in Figure 4.10. In addition, the segments at the end of a sample chromosome are included. Most segments have 1 to 5 outgoing links. A high number of outgoing links can have several causes: Firstly, the segment can be followed by many different variants. Secondly, the segment can occur more often in a path and represent a different region in this context. Then, of course, other links to other segments go out from there. For all segments with more than 90 outgoing links, at least once the segment must occur more often in a path (case 2), as there can only be one variant per sample. The appendix in Figure A.8 shows the incoming links for each segment. This has corresponding causes. As will be explored in more detail later in Section 4.2.3, the segments with the most outgoing links, and thus the most occurrences in all paths, are those with a sequence of only one nucleotide, i.e. A, T, C, or G.

4.2 Results Generated by pan2

Next, data from pan2gene and internal states from pan2index are analyzed. For this purpose, the two tools were called for all protein coding genes in chromosome 1. The two program calls look like this:

```
python3 pan2index -p hprc.gfa -a gencode.v38.annotation.gtf -j -n 1 -t 8 -r 1000
```

```
python3 pan2gene -p hprc.gfa -a gencode.v38.annotation.gtf -j -n 1 -v vg
```

A GRCh38 annotation from gencode was used as the annotation file. This can be downloaded here: https://www.gencodegenes.org/human/release_38.html. A promoter length of 1,000 bp was used. Note that the -j, -a and -n specifications are not necessary for the pan2gene call if no other data was previously generated by pan2index. According to the annotation, chromosome 1 contains 2,049 protein-coding genes. This means that in the end 2049 Mini GFA, VCF, Fasta and MSA files were created.

4.2.1 Mini GFA Statistics

The HPRC pangenom GFA file contained a total of 90 samples. This means that if a gene was found for each sample, 90 paths are expected in the Mini GFA file. Figure 4.13 shows the actual distribution of the number of paths in the Mini GFA files. Most genes contain all 90 paths. But there are also some with less. In total, about 96.78% of the genes contain all paths and about 99.17% contain at least 86 paths. This means that of the 2,049 genes, only 17 contain 85 or fewer paths. Two genes have a particularly low number of paths: OR4F5 contains only 6 paths and OR4F29 only one: the GRCh38 path, which is included in every Mini GFA because it was used as a reference path. These are also the first two protein-coding genes on the chromosome. In the case of ORAF29, the complete gene is contained on a single segment. It is a telomere segment, which to a large extent contains only the gap filler base 'N', as the centomere segments do. These segments have not been aligned with other samples, as previously noted. So it also makes sense that in this case no further paths could be found and extracted. The region where ORAF5 is located seems to have been aligned with at least a few samples.

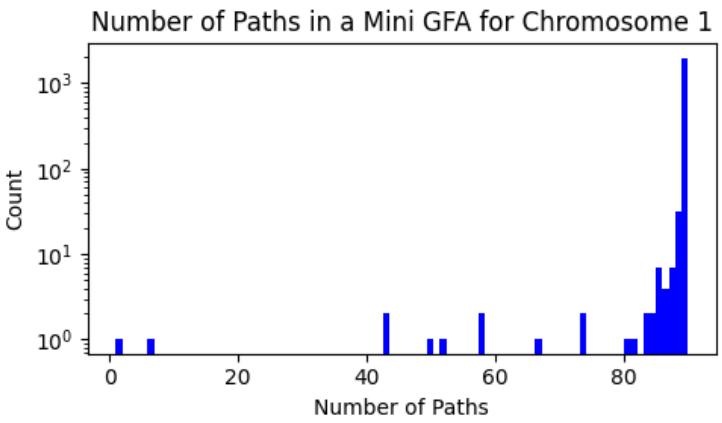

Figure 4.13: **Number of Paths Contained in Mini GFA.**

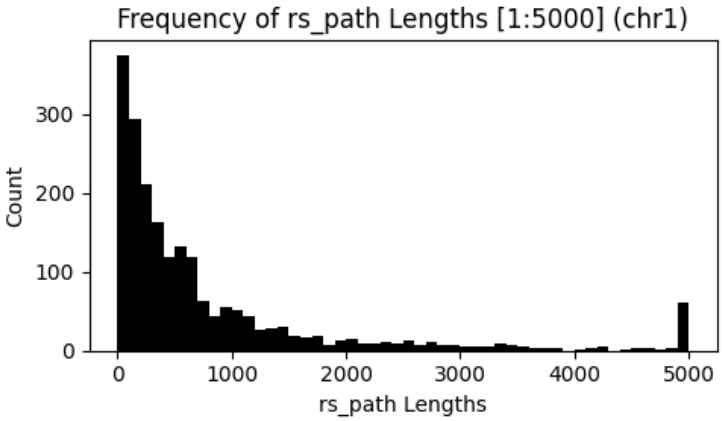

Figure 4.14: **Reference Sub Path Lengths of Chromosome 1.**

In Figure 4.14 the lengths of the `rs_paths` are plotted. This is equivalent to the lengths of the GRCh38 paths in the Mini GFA files. About 350 genes contain less than 100 segments. The frequency of the genes decreases as more segments are included in the gene. The entry for 5,000 segments symbolizes all genes with more or equal than 5,000 segments. The distribution beyond that can be seen in the appendix under Figure A.9. The gene NBPF20 spans the most segments in chromosome 1 with 48,968 segments. This gene contains about 120,000 bp. The longer a gene is, the more segments it is expected to span.

During the Gene Index files Creation in Section 3.2.4 certain flags were set if certain circumstances were present. If the gene is split between several contigs, a '$*$' flag was set and, if no start or end segment could be identified a '?' flag. Paths with a flag are referred to as anomalies. Figure 4.15 shows how many anomalies are contained in a Mini GFA file. Overall, 99.2% of the paths are no anomaly and 90.7% of the Mini GFA files do not contain a single anomaly. It would be expected that more '$*$' anomalies occur the longer the reference path is. Indeed, an increase in contig boundary crossing

paths for longer genes can be seen at the bottom left of the plot. The green marks for short genes and a high number of anomalies are surprising, as the chance of a gene being cross-contig for a sample rises with the length of their `rs_path`. A certain number of '∗' flags can be expected even for shorter genes. It may also be a difficult region to map which is therefore split into several contigs per sample. Potentially, some of that anomalies were wrongly set as cross-contig, as no start and end segment was found. But the reference path of these genes happens to contain many frequently occurring segments, so that the end of the contig was falsely identified as part of the gene. There are also some Mini GFA files that contain a lot of '?' paths. In the gene CELA3A, all paths except the GRCh38 path are marked with this flag. Nevertheless, the suffixes of the paths match to the reference. They start only two segments earlier than the reference path, although many of the paths contain a start segment. The reason for these anomalies could not be identified.

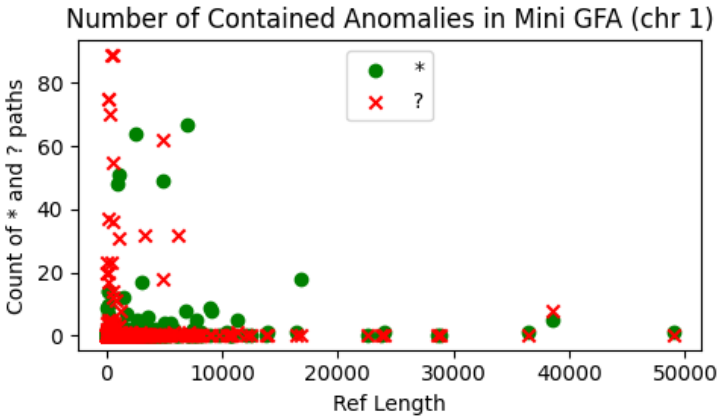

Figure 4.15: **Distribution of Anomalies in Dependence of Reference Path Length.**

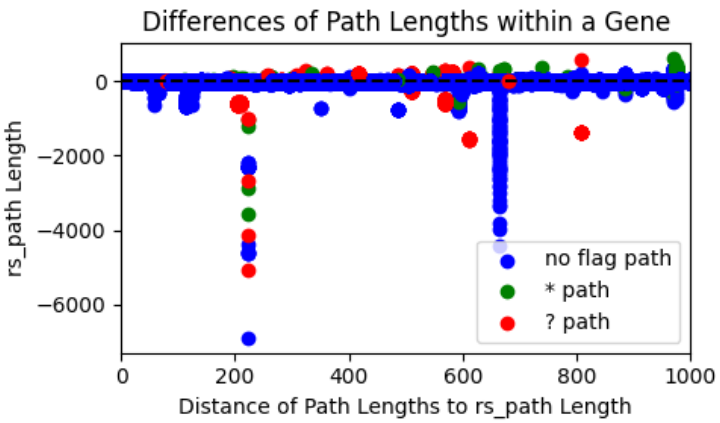

Figure 4.16: **Length Differences in Segments of Sample Paths to Reference Path in Mini GFA.**

Next, the lengths of the paths assigned to a gene are compared (see Figure 4.16). The length of each sample path is subtracted from the length of the reference path. The difference is then plotted. If a path has a positive difference, it means that the reference path is longer than the sample path. If the difference is negative, the reference path is shorter. Some variation is quite normal due to deletions and insertions. It can also occur when a variant appears within a variant. Some deviation is expected for the anomaly paths. Furthermore, the deviation is expected to grow for longer genes. In the plot, only the genes with less than 1,000 segments are shown for a better overview. Again, the complete plot can be found in the appendix under Figure A.10. It can be seen that there are paths that surprisingly long compared to their reference path. Vertical lines of paths are formed in the plot. Such a line represents a gene. Paths that are 3,000 segments or more longer than their reference path are very unlikely. It can be quite safely assumed that incorrect sample paths have been extracted for these genes. For example, for the gene AMY2A, a path is found that is approximately 7,000 segments longer than the reference path, which is only 224 segments long. A mutation that increases or decreases the path length of a gene by more than 3,000% is not very plausible. The extended version of the plot in the appendix also shows that this disproportionate dispersion increases for even longer genes. Some paths identified

as genes for the gene TTC34 contain up to 200,000 more segments than their reference path. Since these much too long paths are labeled as no flag paths, one explanation for this behavior is most likely: No start and/or end segment was found in the initial traversing of the best-scored subpath and incorrect occurrences of the start and/or end segment were found in the subsequent traversing of the contig path (Section 3.2.4). These paths are not marked with a flag and contain highly erroneous information about the gene. It should also be noted that any cross-contiguous '$*$' path longer than the reference path is also very unlikely. The green paths in the negative area of the y axis in Figure 4.16 represent the incorrectly labeled green paths on the left side of the plot in Figure 4.15. However, '$*$' paths that are shorter than the reference path are realistic. Apart from that, some '?' paths are displayed that contain too many segments. Overall, 0.88% of the paths have a deviation greater than 500 segments compared to the reference path.

As last point on the Mini GFA statistics, figure A.11 in the appendix is referred to. There, the number of contained segments in a mini GFA file was plotted against the number of contained links. The previously calculated constant 1.4 for the chromosomes can also be calculated from the Mini GFA files. The plot approximates a linear function.

4.2.2 MSA, Fasta and VCF File Statistics

Next, the other output files besides the Mini GFA files are analyzed by pan2gene. For the MSA files, the length of each multiple sequence alignment is plotted in Figure 4.17. All aligned sequences in the MSA have the same length. Thus, an MSA length can be determined for each gene. The length of an MSA, according to the method used in Section 3.3.4, depends on three things. The length of the paths, the length of the segment sequences in the paths and the number and length of variants. So genes with a low reference path length are expected to have a shorter MSA length as long as they do not contain particularly long segment sequences or particularly large variants. The red line in the plot represents the median. 50% of the MSAs are longer than 28,000 bp. MSAs with greater length are less frequent The gene TTC34 has by far the longest MSA with a $2 * 10^7$ bp long MSA. This gene has already been noticed in Figure A.10 because of the much too long paths (it is the gene with the ref path length 39,000). Since it contains such long paths, it also makes sense that the MSA is so long.

In Figure 4.18 the number of different haplotypes per gene is shown. This information can be extracted from the Fasta file as well as from the additionally generated haptab file. The red line again symbolises the median, which is 7. Most genes have 1-4 haplotypes but starting from about 20 haplotypes the plot is uniformly distributed. There are also genes like OMA1 that contain 90 different haplotypes. So does this mean that each of the 90 samples has a different protein variant for this gene? No, apart from the fact that a mutation of a nucleotide can still translate to the same amino acid, the identified gene sequences also contain introns and a 1,000 bp long promoter region. But non-coding variants can be important as well.

Next, the VCF files are evaluated. In Figure 4.19, the number of variants of a gene is plotted against its reference path length. A mark in the plot represents a gene. Most genes contain about 5 to 2000 variants. The distribution of genes can be roughly divided into two groups. Group 1 is on the left of the plot and contains under 1000

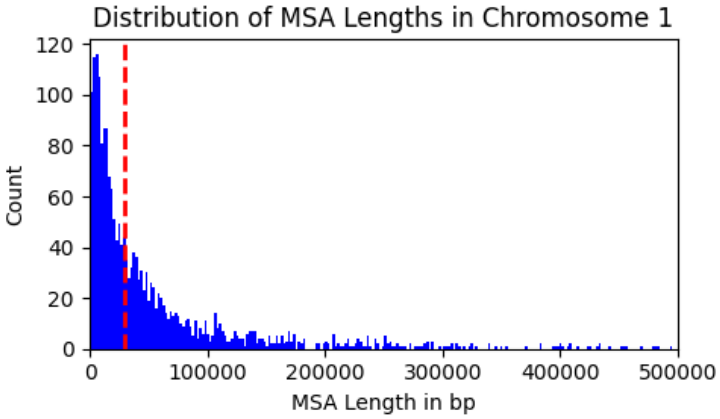

Figure 4.17: **Distribution of MSA Lengths in Chromosome 1.**

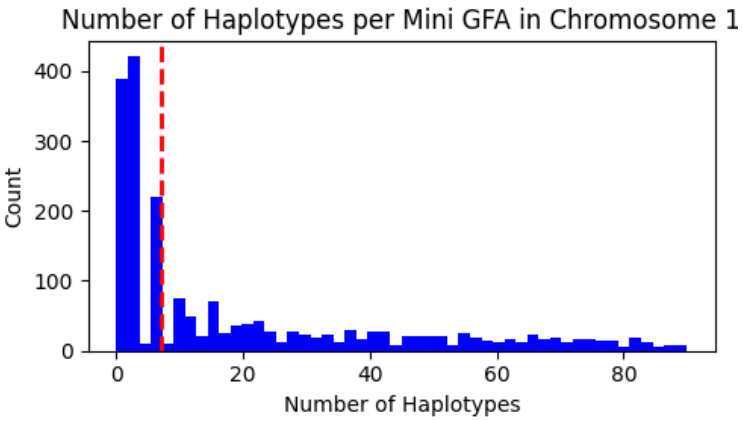

Figure 4.18: **Number of Haplotypes per Gene in Chromosome 1.**

variants. These genes probably contain fewer different haplotypes and are relatively well conserved across the population. The number of variants is quite independent of the length of the reference path. Group 2 of the genes is on the right of the plot and contains 1000 or more variants. It can be seen here that the number of variants increases as the length of the reference path increases. These genes are not as well conserved as those in group 1. The genes on the far right of the plot are genes such as DAB1 or KAZN, which contain 89 or 90 haplotypes. Interestingly, the much too long paths from gene TTC34 do not impact on a particularly large number of variants. The gene at the bottom left is ORAF29, which contains only one path and one segment and thus not a single variant.

Figure 4.20 shows how long the longest variant is for each gene. For some further processing of the data, it may be important to know how long variants are to be expected. The value 400 again stands for all higher values added up. About 50 genes contain no variants at all and 20 only variants that are 1 bp long. In addition to the peak for very short genetic variants, there is another peak in the interval [310, 350]. About 140 genes contain a variant longer than 400 bp. The longest variant is part of the gene TTC34 and is about $4 * 10^5$ bp long. So, when creating the VCF files, 'vg deconstruct' creates one large variant instead of many small ones for very different paths. Overall, 82.03% of all variants of chromosome 1 are only 1 bp long and only 4.21% are longer than 5 bp.

4.2.3 Additional Data from Internal pan2index States

When executing pan2index, information about the internal states of the tool are provided by setting the -s parameter. These give a further, deeper insight into the data. The data is still based on chromosome 1. Figure 4.21 gives an insight into the dictionary `seq_pos` (see Section 3.2.2). The plot shows how often certain segments occur in all sample paths. The graph starts with 10^3 segments that occur only once in the sample paths. Then the plot increases until it peaks at 89. Almost 10^6 segments occur this often in the sample paths. These segments are best suited to identify a gene, as

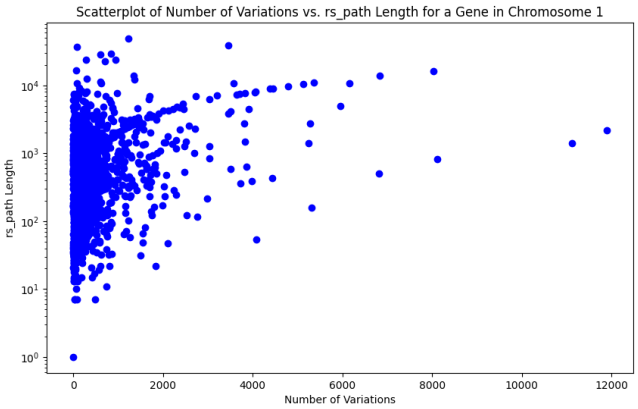

Figure 4.19: **Comparison of Number of Variants with Ref Path Length.**

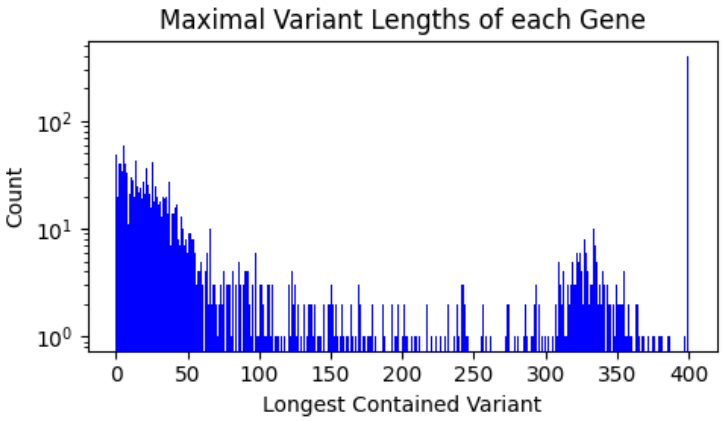

Figure 4.20: **Distribution of Longest Variants for each Gene.** All variants longer than 400 bp are represented by the value 400.

most of them are expected to be contained exactly once by each sample. The peak is not at 90 because the reference path is not included in `seg_pos`, since the considered segments have been extracted from it and are no longer found in it. From this point on, the plot has a certain periodicity with length 89. The frequency of the segments decreases rapidly and then rises to a new peak. This structure is formed because many segments are well conserved in the sample paths. Samples that occur 178 times are contained in two places in the chromosome and therefore occur twice for each sample. The higher the period, the more this structure blurs with noise. The plot only shows segment frequencies up to 400, but there are also segments that occur even more frequently. With 68,268 occurrences, segment 145263 is the most frequent segment in chromosome 1 and contains the sequence 'C'. Each of the very frequent segments contains only one nucleotide as a sequence. A pattern of which 1bp long segment is reused at the pangenome creation and which is not cannot be identified.

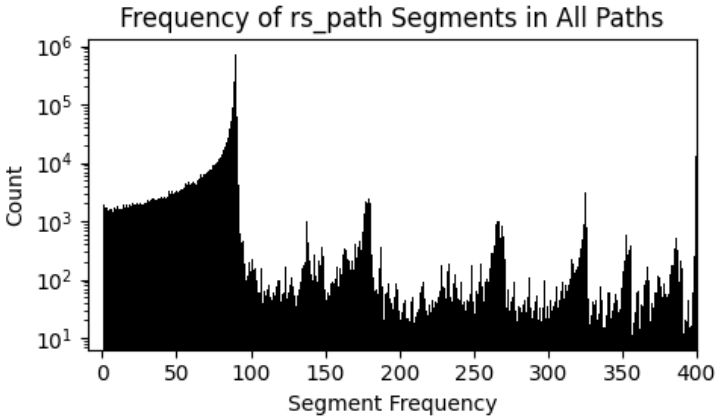

Figure 4.21: **Frequency of rs_path Segments in all Paths (seg_pos).**

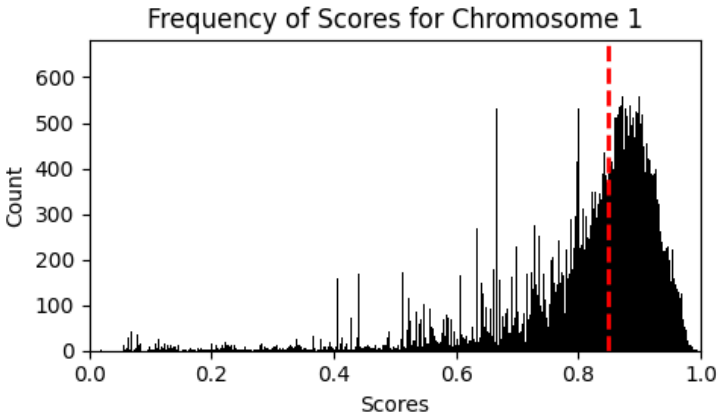

Figure 4.22: **Scores of Extracted ss_paths.**

In Section 3.2.4, a score is calculated for each `merged_interval` for each contig of a

sample and each gene. The region with the best score is then identified as the gene. In Figure 4.22 the distribution of these best sample scores is shown. The Jaccard index, which always lies in the interval [0,1], was used as the scoring method. The higher the score, the more certain the region contains the gene. Most of the scores have a high value, showing high credibility that the contig is mapping to the gene. The median is 0.85, but there are also very low scores. 16.2% of the scores are lower than 0.75. The lower the score, the higher the probability of a misidentified gene, unless the gene is very small. In an improved version of pan2index, consideration could be given to sorting out regions with too small scores for their `rs_path` length.

5 Related Work

In addition to the HPRC pangenome, other methods have been developed to provide access to pangenome data.

5.1 PanGenie

PanGenie (for Pangenome-based Genome Inference) offers a different approach to gain insight into genome variation within a species [21]. PanGenie not only constructs a pangenome graph from multiple individual genomes, it also genotypes a wide spectrum of variants. Similar to the HPRC Pangenome (pggb), a series of haplotype-resolved genome assemblies is entered as input, for example as a Fasta file. A pangenome graph is then built from these. However, a node does not contain a segment with variable sequence length, but a k-mer with a fixed sequence length. Bubbles in the graph represent a variant and a path represents a haplotype. Short-read sequencing data of an uncharacterized sample which is also subdivided into correspondingly long k-mers can then be mapped onto this graph. For each node in the graph, the number of k-mers mapping against it is counted, which then determines the two haplotypes of the input reads by following paths with high k-mer counts. An overview of the algorithm used by PanGenie can also be found in the appendix under Figure A.12. By using the k-mers approach, PanGenie is performing worse for repeat-rich regions. Nonetheless, PanGenie provides a good method for association and genotyping studies for all variant types when high-quality reference assemblies are available. [21]

5.2 gnomAD

The Genome Aggregation Database (gnomAD) is presently recognized as the largest and most extensively utilized public repository of population variation, derived from harmonized sequencing data [22]. Building upon the foundation laid by the Exome Aggregation Consortium (ExAC), gnomAD encompasses genetic variation data from 76,156 whole genomes in the GRCh38 set obtained from unrelated individuals who were sequenced as part of diverse disease-specific and population genetic studies, making it an extensive and comprehensive resource [7, 23]. The dataset can be accessed via the gnomAD browser, a web-based platform available at https://gnomad.broadinstitute.org/. gnomAD uses a different pangenomic approach than the HPRC: the direct utilization of an online database. With its focus on pangenomics, gnomAD offers a unique perspective by incorporating diverse genomic sequences beyond the traditional reference

genome. This enables a more accurate representation of human genetic diversity and a better understanding of rare and common genetic variants across populations. By integrating data from multiple sources and sequencing projects, gnomAD offers a vast catalog of genetic variants, including single nucleotide variants (SNVs), insertions and deletions (indels), and structural variants. These variants are annotated with valuable information such as allele frequencies, functional impact predictions, and population-specific metrics, allowing researchers to explore the functional and clinical implications of genetic variation. Furthermore, gnomAD gives information about the ethnic distribution of the variants. However, unlike the HPRC pangenome, the sequences in gnomAD are not aligned. [23, 24]

5.3 GRAF

Another alternative approach to pangenomic analysis and variant calling is provided by Seven Bridges' tool GRAF [25]. GRAF also uses a directed acyclic graph as the structure for the pangenome, but unlike pggb from HPRC and PanGenie, it does not use multiple haplotype-resolved genome assemblies to construct it. Only one reference genome is used, which acts as the backbone of the graph. This is then expanded with variants from sources such as the 1K Genomes Project and Simons Genome Diversity Project [12, 26]. Again, a path through the graph represents a haplotype. Input reads can be mapped onto this graph, resulting in graph-based alignments and variant calls. In the pangenome graph used, as in the HPRC and gnomAD, both small variants (indels/SNVs) and larger structural variants are included. In addition to the pangenome graph, other population-specific graphs are provided. They increase the sensitivity to the respective population and are recommended for specific population studies. Overall, GRAF provides mapping and genotyping of sample reads, variant calls and visualizations of multiple graph references. However, this tool has not been widely used and thus evaluated in scientific studies. [25]

6 Discussion and Outlook

The main objective of this thesis was to unravel the structure of the HPRC pangenome for further usage and to develop a tool that can extract genetic data from this pangenome. This was achieved with the pan2 workflow.

However, the pan2 workflow has some limitations. For very large genes such as TTC34, incorrect gene sequences are returned and incorrect flags are set. This problem occurs when none of the start or end segments of the reference path are contained or when start and end segments are repeated in the gene itself or in the region around the gene. In addition, it is not possible to extract gene data from the samples if the gene is located on a telomere, centomere or gap segment such as ORAF29. Sometimes one or two too many segments are read out, as in the case of the gene CELA3A, but it is marked with a flag. Furthermore, the algorithm of the MSA is only reliably correct if segments are not contained multiple times in the same gen path. This problem occurs because if the segment is not in a variant, it is only checked whether the segment is in the sample path, not where in the path. This problem is not easy to circumvent, since one can rely on the previous alignment of the sequences of the same segment, but not on the alignment of the segment IDs in the gene paths. Since these paths can contain more than 10,000 segments (see Figure A.9), this NP-Complete problem would not be computable. It is expected that the probability for a multiple occurring segment that is not included in a variant and may be deleted at one position is low, but increases with growing gene length. In all other cases, the extracted data is correct.

The algorithm of pan2index could undergo some improvements in a future version. In relation to the length of the `rs_path` of a gene, a minimum and maximum length for `ss_paths` could be defined. This could prevent, paths that are far too long from being identified as genes, as is the case for TTC34. Another approach for improvement is to score different segments differently. In Figure 4.21 it becomes clear that a large number of segments is only contained in the chromosome as many times as the number of samples. Segments in `seq_pos` with a frequency close to the number of samples can be considered a stronger indication for the genetic identification of a region than segments that occur much more frequently. In addition, the best-scoring regions of a sample could be sorted out if their score is below a certain threshold in relation to the length of the gene. At least they could be marked with an additional flag. In addition, genes whose initial region has been extended by an additional search in the contig path can be marked with a flag as well. Furthermore, a potentially better value for `span` can be elaborated.

An alternative approach to extracting gene sequences from the pangenome could be to map the sequence of a gene's `rs_path` to the entire graph using tools like vg and then extract the most frequently scored paths. There are certainly more, alternative

approaches to this task.

The analysis of the HPRC pangenom GFA file has shown that the paths take up the most space in the file. A path line can be over 10^8 bytes long. The text-based GFA format does not seem to be optimal for storing a pangenome. As text, each nucleotide is stored in one byte. With 'N' as the fifth option, however, only three bits would be necessary. In addition, repetitive segments with identical sequences are created. GFA files are compressed with standard data compressors like gzip [15]. The paths in the GFA are very similar to each other, which makes the file highly repetitive. However, for gzip compression, no window size larger than a few megabytes is used. Thus, because the path lines are so long, they are rarely in the same window, so the similarities cannot be compressed. The more sample assemblies are added to the pangenom GFA, the greater the extent of these problems. The GBZ file format could solve this problem [15]. In contrast to the GFA format, the GBZ format is a path-based format that stores the long similar paths space efficiently. Nodes and edges are dynamically created only during usage, instead of being statically written into the file. The GBZ format is also supported by vg and can be transformed into a GFA format as well as built from a GFA file. Currently, only paths can be extracted from the GBZ format, but it is expected that the format will evolve. There will definitely be an update of the file format of the pangenome. [15]

With the pan2 workflow, gene sequences can now be extracted from the pangenome and written into mini GFA files. In addition, VCF files with the variants, Fasta files with the haplotypes and MSA files for each gene are created. These files contain the human diversity of the pangenome. They can be used for further studies and analyses. With the Mini GFA files created, tools such as vg or odgi can be used, for example, to create visualisations of the graph structure and the variants. Furthermore, the Mini GFA files can be augmented with additional gen data. [8, 9]. The data can also be imported into gene databases, improving the overall gene-disease association and the general understanding of the human genome. As soon as the HPRC publishes a new version of the pangenome with more genomes included, the gene data can be immediately extracted and updated with pan2 or an improved version of the tool. It should be noted, however, that although a pangenome represents human diversity better than a linear reference genome, it is still only an approximation.

In addition, the generated data will be transferred to ISAR (Isoform Structure Alignment Representation). ISAR aligns all homologous genes for a gene, therefore paralogous genes from the same species and paralogous genes from different species. The aim is to build up a database in this way and to learn about evolutionary changes in the structure of genes. With the help of the pangenome generated data, human variants can now be better estimated, which will hopefully lead to a better understanding of the general structure and variance of a gene.

Bibliography

[1] T. Hawkins *et al.*, „Initial sequencing and analysis of the human genome“, *nature*, vol. 409, no. 6822, pp. 860–921, 2001.

[2] National Human Genome Research Institute (NHGRI). „National human genome research institute: Genetics vs. genomics fact sheet“. (2018), [Online]. Available: `https://www.genome.gov/about-genomics/fact-sheets/Genetics-vs-Genomics` (visited on 08/11/2023).

[3] V. A. Schneider *et al.*, „Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly“, *Genome research*, vol. 27, no. 5, pp. 849–864, 2017.

[4] D. G. Augusto *et al.*, „A common allele of hla is associated with asymptomatic sars-cov-2 infection“, *Nature*, pp. 1–9, 2023.

[5] R. E. Peterson *et al.*, „Genome-wide association studies in ancestrally diverse populations: Opportunities, methods, pitfalls, and recommendations“, *Cell*, vol. 179, no. 3, pp. 589–603, 2019.

[6] T. Wang *et al.*, „The human pangenome project: A global resource to map genomic diversity“, *Nature*, vol. 604, no. 7906, pp. 437–446, 2022.

[7] L. Koch, „Exploring human genomic diversity with gnomad“, *Nature Reviews Genetics*, vol. 21, no. 8, pp. 448–448, 2020.

[8] E. Garrison *et al.* „Variation graph data structures, interchange formats, alignment, genotyping, and variant calling methods“. (2014), [Online]. Available: `https://github.com/vgteam/vg` (visited on 07/11/2023).

[9] A. Guarracino, S. Heumos, S. Nahnsen, P. Prins, and E. Garrison, „Odgi: Understanding pangenome graphs“, *Bioinformatics*, vol. 38, no. 13, pp. 3319–3326, 2022.

[10] S. Aganezov *et al.*, „A complete reference genome improves analysis of human genetic variation“, *Science*, vol. 376, no. 6588, eabl3533, 2022.

[11] S. Nurk *et al.*, „The complete sequence of a human genome“, *Science*, vol. 376, no. 6588, pp. 44–53, 2022.

[12] N. Siva, „1000 genomes project“, *Nature biotechnology*, vol. 26, no. 3, pp. 256–257, 2008.

[13] D. Crysnanto, A. Leonard, and H. Pausch, „Comparison of methods for building pangenome graphs“, in *Proceedings of 12th World Congress on Genetics Applied to Livestock Production (WCGALP) Technical and species orientated innovations in animal breeding, and contribution of genetics to solving societal challenges*, Wageningen Academic Publishers, 2022, pp. 1066–1069.

[14] E. Garrison *et al.*, „Building pangenome graphs“, *bioRxiv*, pp. 2023–04, 2023.

[15] J. Sirén and B. Paten, „Gbz file format for pangenome graphs“, *Bioinformatics*, vol. 38, no. 22, pp. 5012–5018, 2022.

[16] E. Garrison *et al.* „Vg: Tools for working with variation graphs“. (2014), [Online]. Available: `https://vg.readthedocs.io/en/latest/index.html#` (visited on 08/05/2023).

[17] L. Wang and T. Jiang, „On the complexity of multiple sequence alignment“, *Journal of computational biology*, vol. 1, no. 4, pp. 337–348, 1994.

[18] F. Sievers and D. G. Higgins, „Clustal omega“, *Current protocols in bioinformatics*, vol. 48, no. 1, pp. 3–13, 2014.

[19] M. A. Wilson, „The y chromosome and its impact on health and disease“, *Human Molecular Genetics*, vol. 30, no. R2, R296–R300, 2021.

[20] P. B. Talbert and S. Henikoff, „What makes a centromere?“, *Experimental cell research*, vol. 389, no. 2, p. 111 895, 2020.

[21] J. Ebler *et al.*, „Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes“, *Nature genetics*, vol. 54, no. 4, pp. 518–525, 2022.

[22] S. Gudmundsson *et al.*, „Variant interpretation using population databases: Lessons from gnomad“, *Human mutation*, vol. 43, no. 8, pp. 1012–1030, 2022.

[23] M. J. Daly *et al.* „Gnomad: Genome aggregation database“. (2014), [Online]. Available: `https://gnomad.broadinstitute.org/` (visited on 08/01/2023).

[24] K. J. Karczewski *et al.*, „The mutational constraint spectrum quantified from variation in 141,456 humans“, *Nature*, vol. 581, no. 7809, pp. 434–443, 2020.

[25] M. Marinkovic. „Seven bridges graf: Graph-based workflows for next-generation sequencing data“. (2020), [Online]. Available: `https://www.sevenbridges.com/graf/` (visited on 08/11/2023).

[26] S. Mallick *et al.*, „The simons genome diversity project: 300 genomes from 142 diverse populations“, *Nature*, vol. 538, no. 7624, pp. 201–206, 2016.

A Appendix

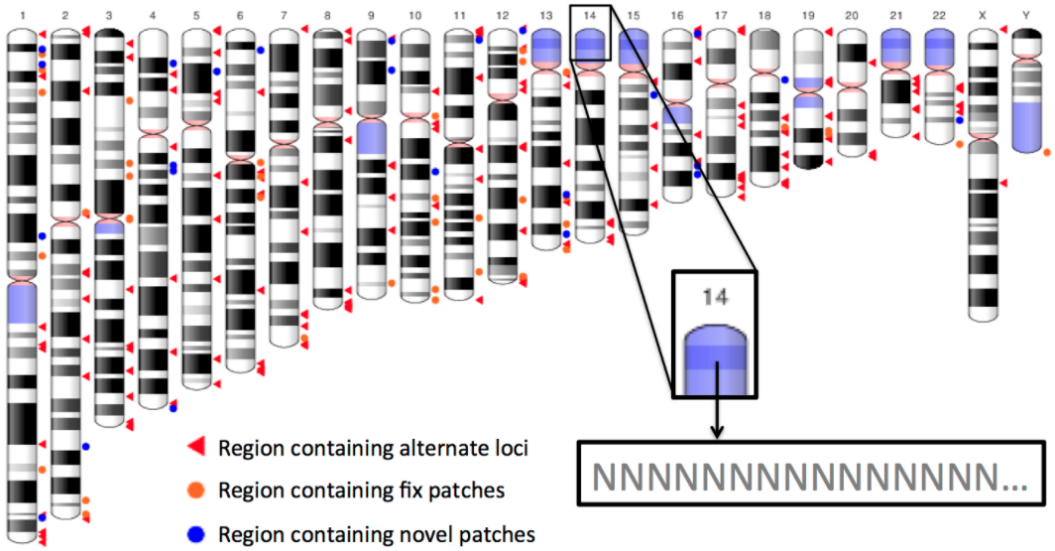

Figure A.1: **grch38 Ideogram.** This image is from the Genome Reference Consortium website. It shows GRCh38.p7. The blue regions contain unspecified nucleotides replaced by 'N'. https://gatk.broadinstitute.org/hc/en-us/articles/360035890951-Human-genome-reference-builds-GRCh38-or-hg38-b37-hg19

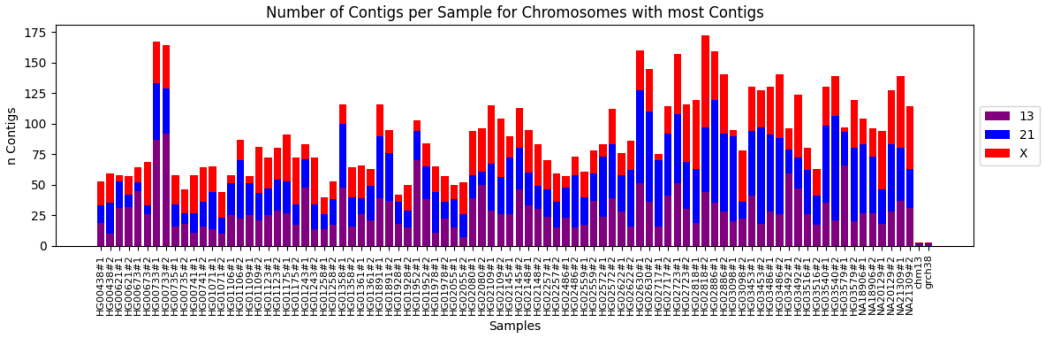

Figure A.2: **Number of Contigs per Sample for Chromosomes with most Samples.** Not every sample has as much contigs in this chromosomes as some have. However the samples with the most contigs here are also the samples with the most overall contigs seen in Figure 4.7

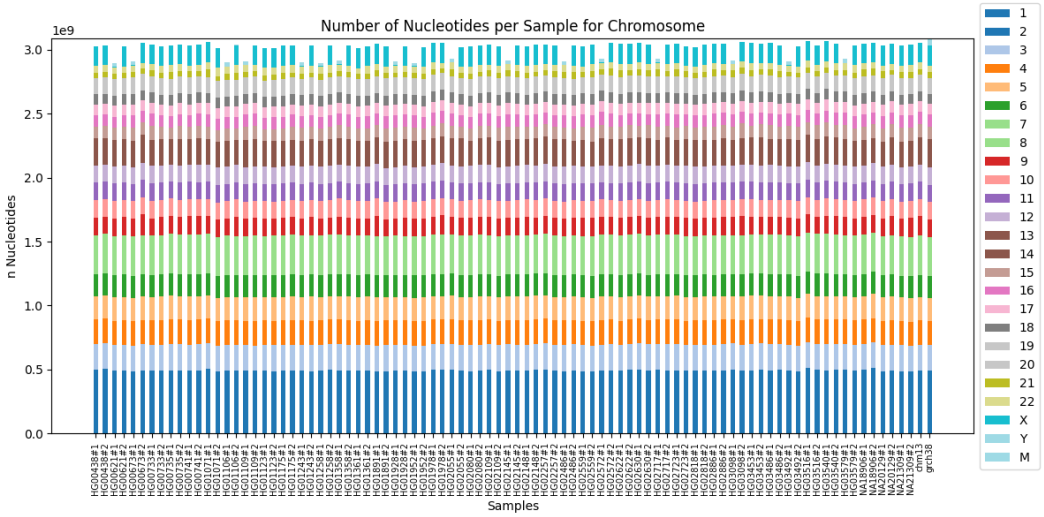

Figure A.3: **Number of Nucleotides per Sample for All Chromosomes.** Every sample is about $3 * 10^9$ bases long. The samples containing a Y chromosome and thus no 'real' X chromosome have a slightly shorter overall sequence.

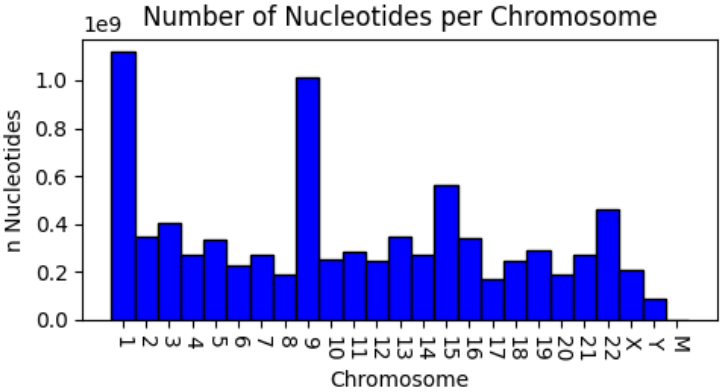

Figure A.4: **Number of Nucleotides per Chromosome.** Chromosomes with many segments contain also more nucleotides.

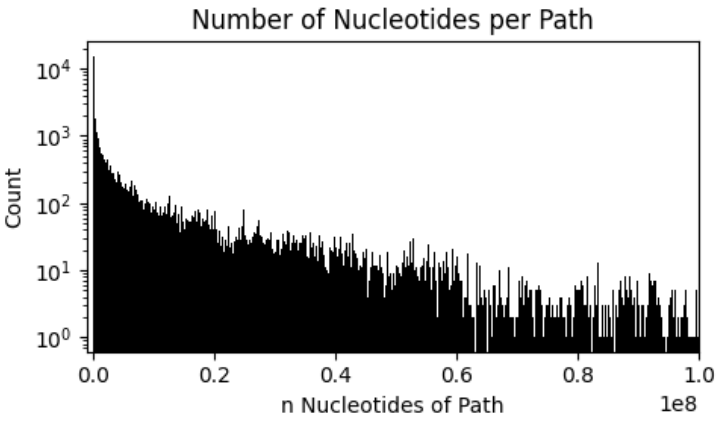

Figure A.5: **Number of Nucleotides per Path.** The longest path is a reference genome path with a length of 248956422.

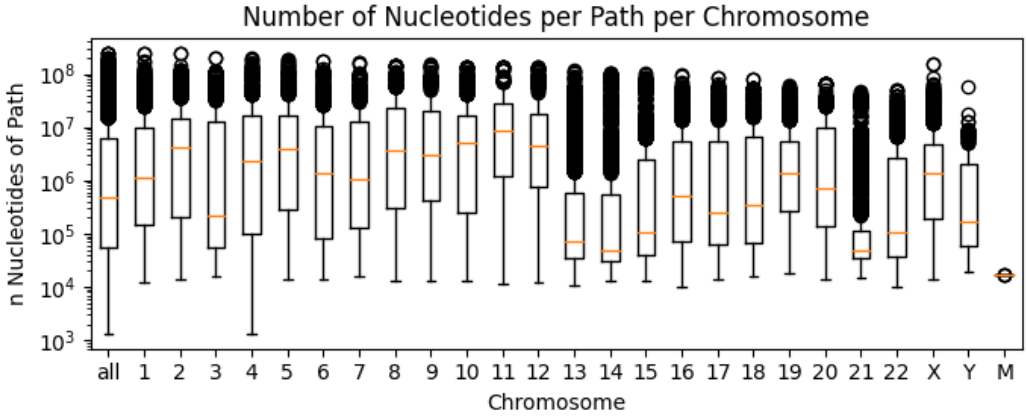

Figure A.6: **Number of Nucleotides per Path per Chromosome as Boxplot.** Chromosomes with more paths contain on average fewer nucleotides per path.

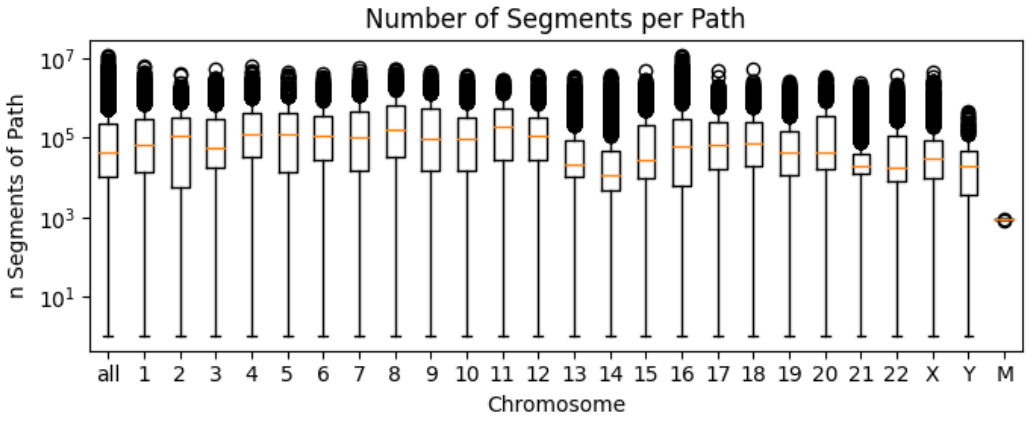

Figure A.7: **Number of Segments per Path per Chromosome as Boxplot.** The average number of segments per path is more constant across the chromosomes than for the nucleotides.

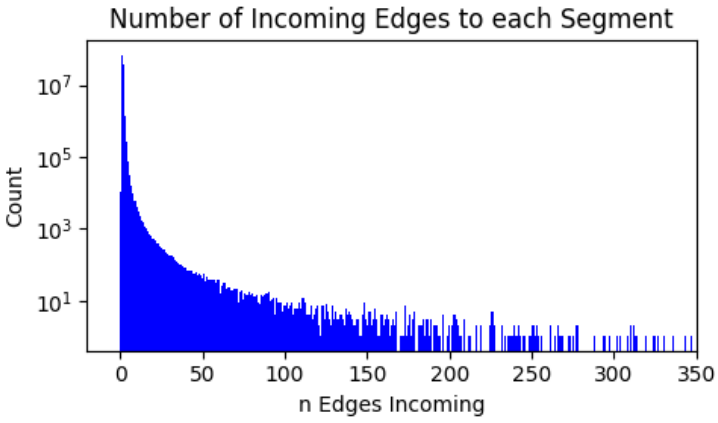

Figure A.8: **Number of Incoming Links for each Segment.** Chromosomes with many segments contain also more nucleotides.

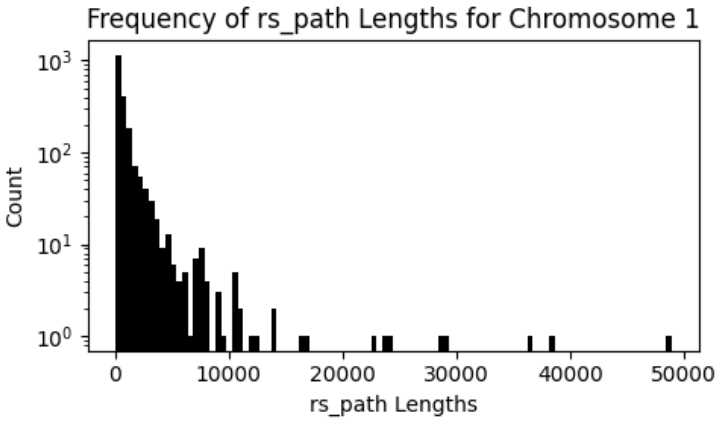

Figure A.9: **Reference Sub Path Lengths Complete.** All lengths of the reference paths in the Mini GFA files created in Chromosome 1.

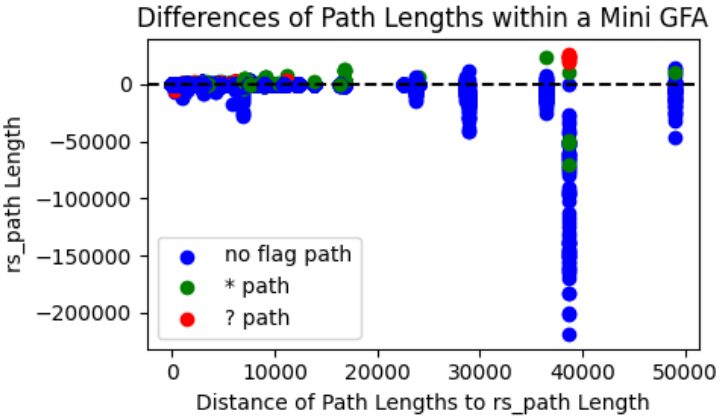

Figure A.10: **Length Differences of Sample Paths to Reference Path in Mini GFA Whole.** Very long genes also show a very large deviation in the lengths of the paths. The algorithm seems to be error-prone for such large genes.

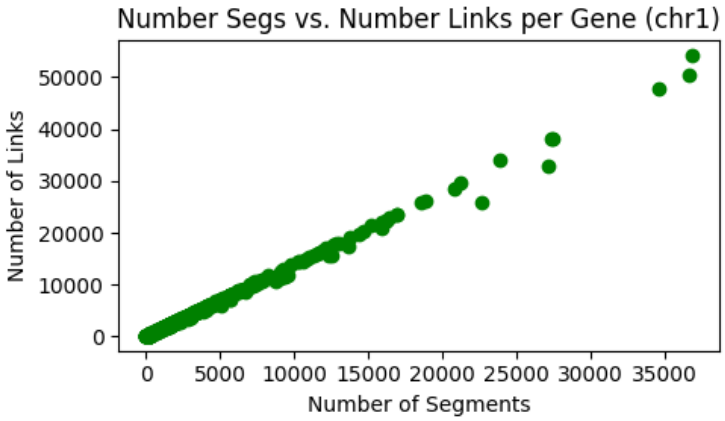

Figure A.11: **Comparision of Number of Segmens and Number of Links in a Mini GFA File.** The plot approximates the function f(x) = 1.4x. The slope depends on the number of samples.

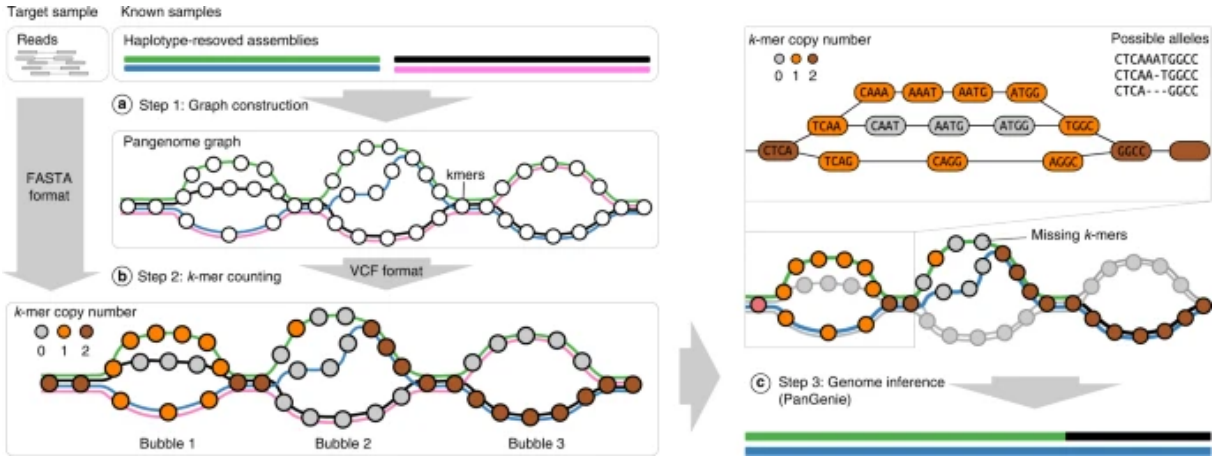

Figure A.12: **PanGenie Algorithm Overview.** The nodes in the graph represent k-mers, a bubble represents alleles and a path through the graph is a haplotype. Using haplotype-resolved genome assemblies and reads from an uncharacterised sample, the sample is genotyped. This Figure is adapted form the paper 'Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes' [21].

Algorithm 1: `find_genes_in_sample_paths(pangenome_gfa, genes, paths, span, max_val_size)`

```
Result: Extract gene subpaths from sample paths and write the into Gene
        Index files
begin
    for chrom in genes do
        rs_paths = generate_rs_paths(chrom);
        seq_pos = segment_indexing(rs_paths);
        for gene in chrom do
            bucket = extract_bucket(seq_pos, gene);
            rs_path = rs_paths[gene];
            ref_sets = fill_ref_sets(rs_path);
            for sample in bucket do
                best_val = None; best_score = 0;
                for contig in sample do
                    intervals = transform_bucket_pos(bucket, span);
                    intervals.filter(max_val_size);
                    merged_intervals = merge(intervals);
                    for merge in merged_intervals do
                        score = calculate_score(merge);
                        if score > best_score then
                            best_val = merge; best_score = score;
                contig_path = read_best_val_path_from_gfa(best_val,
                 pangenome_gfa);
                ss_path = contig_path[best_val];
                start_segs, end_segs = get_start_and_end( rs_path);
                if check_contig_strand(ref_sets, ss_path) then
                    switch_start_and_end_segs();
                check_if_ss_path_start_contains_start_seg();
                if not then
                    search_in_contig_path_backwards_from_ss_path_start();
                check_if_ss_path_end_contains_end_seg();
                if not then
                    search_in_contig_path_from_ss_path_end();
                ss_path = cut_at_found_posisions(ss_path);
                set_flags(ss_path);
                ss_path = reverse_if_necessary(ss_path);
                save_ss_path;
            write_gene_index_file(gene, ss_paths);
```

Figure A.13: **Pseudo Code for pan2index after Identification of Start and End in the Reference Path.** The genes dictionary is divided into chromosomes and provides calculated data from Section 3.2.1