Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: 29 March 2019
Manuscript Category

OXFORD

Genome analysis

Annotation transfer within pangenomes

Nicky Faber [1,*]

[1] Bioinformatics group, Wageningen University, Wageningen, Droevendaalsesteeg 1 6708PB, The Netherlands

* To whom correspondence should be addressed.

Received on 22 March, 2019; revised on 28 March, 2019; accepted on 28 March, 2019

Abstract

Motivation: Comparative genomics is a field of research that tries to link genotype to phenotype by comparing many genomes of related organisms. With the help of genome annotation, these analyses can lead to the discovery of biological processes that influence a certain phenotype, which could be exploited in many fields such as agriculture, microbiome analysis and health and disease. However, because many similar genomes are used in comparative genomics, it would be more efficient to transfer already curated annotations from present genomes to a new genome rather than to manually curate each annotation separately. The structure of a pangenome, a graph based database that contains many genomes of a population, can facilitate this transfer of annotations very well, and even allows the transfer of annotations to happen comparatively in the future.

Results: Pairwise annotation transfer was implemented as a built in functionality in pangenome tool Pantools as a starting point for comparative annotation transfer. It is able to efficiently and cheaply transfer annotation from a reference genome to a newly sequenced genome. This can be done very well between assembly versions and strains or accession of many different species, and even between different species to a degree. Pantools can transfer the coding sequences of genes with a very high-quality results, but the transfer of the UTRs of genes doesn't yield high-quality results. It was shown that even pairwise annotation transfer yields a relatively complete annotation for the newly sequenced genome, and that it is computationally feasible to transfer many genes in big genomes. However, it is not yet scalable to transfer annotation within very large pangenomes. Now that a framework is in place for pangenomic annotation transfer, the door has been opened for future implementations such as comparative annotation transfer, the integration of RNA-seq data for more precision and the addition of more annotation features such as breeding markers or tRNAs among many others.

Availability: https://git.wur.nl/bioinformatics

Contact: sandra.smit@wur.nl

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Keywords: Annotation transfer, pangenomes, graph based genomes, comparative genomics, annotation

1 Introduction

Until the development of next-generation sequencing technologies, it took a huge effort to sequence a single genome (Fleischmann *et al*., 1995; Venter *et al*., 2001). Nowadays, whole-genome sequencing is routinely done on a large scale in many fields such as agriculture (Li *et al*., 2014), microbiome analysis (Gilbert *et al*., 2014) and health and disease (Siva, 2008). This wealth of information has opened up the door for comparative genomics, a field of research that tries to link genotype to phenotype by comparing the genomes of different organisms. High quality genome annotations are vital to give biological context when these links are found, and this information can be applied, for example, to improved breeding of crops and livestock in agriculture, to gain more knowledge on gut and soil microbiomes, or to develop personalized medicine in health and disease, among many others. Although this field of research is extremely relevant and broadly applicable, it still has some limitations. One of these limitations is the fact that producing a high-quality annotation is a time-consuming and labour-intensive process.

Annotation for newly sequenced genomes is usually done using *de novo* tools such as MAKER2 (Holt & Yandell, 2011) or BRAKER1 (Hoff *et al*., 2015). However, these predict genes with a rather low sensitivity and specificity, both only about 70% (Hoff *et al*., 2015), which necessitates a lot

of manual curation to produce a high-quality annotation. Considering the large amount of genomes used in comparative genomics and the fact that these genomes are all related to a certain extent, people have started looking to a more efficient way to obtain high-quality annotations: annotation transfer. By transferring a well-curated annotation onto another genome, a higher quality annotation can be produced immediately (Otto *et al*., 2011; Fiddes *et al*., 2018). Besides this advantage, it would also be possible to transfer homemade annotation features such as plant breeding markers to another genome. Although annotation transfer is a good strategy, there are some flaws in the design of current tools.

Two generally applicable tools for annotation transfer exist, namely RATT (Rapid Annotation Transfer Tool) (Otto *et al*., 2011) and CAT (Comparative Annotation Toolkit) (Fiddes *et al*., 2018). Other tools are specific to certain organisms or are only made to work on genome assembly versions as annotation liftover. RATT works by finding synteny blocks between genomes to transfer annotation in conserved regions, and CAT uses a multiple sequence alignment and optional RNA-seq data. The problem with RATT is that it creates an internal error while running and it can only be used on .embl format, which makes using it cumbersome at best. CAT is nontransparent in the way it integrates its data and it is extremely difficult to install because of third party software dependencies, which limits its practicality. Besides these inconveniences, a major disadvantage of both tools is that although they can use multiple annotations as reference to transfer from, they can only be used on one genome at a time. Even though annotation transfer can produce high-quality genome annotations with less manual curation than *de novo* tools, it would still be time-consuming and labour-intensive to use on the many genomes used in comparative genomics. The solution to this problem is to make annotation transfer catch up with the newest advancement in comparative genomics: pangenomes.

A pangenome is a data structure that contains all genomes of a population (The Computational Pan-Genomics Consortium, 2016). The reason pangenomes were developed for comparative genomics is because the increasing amount of genomic data showed that a single reference genome cannot contain all variation in a population (Tettelin *et al*., 2005; Zhao *et al*., 2018). This meant that pairwise comparisons had to be done for all combinations of genomes, which is not practical or feasible. As having all genomes in one pangenome greatly improves comparative genomics analyses, both in quality and ease of use, it also has the potential to solve the problems of annotation transfer. A pangenome tool that provides a natural framework for annotation transfer is Pantools (Sheikhizadeh *et al*., 2016), as similar genomic regions are compressed. Besides this, annotations can be added to the pangenome and new genomes can be added easily. This framework allows for annotation transfer to multiple genomes at once, which is fast and convenient, and furthermore, it allows for multiple genomes to serve as a reference, which is important, as one genome might not have all genes that are present in a population (Tettelin *et al*., 2005; Zhao *et al*., 2018). Therefore, it is important that annotation transfer is implemented in a pangenomic framework.

Here we present the implementation of annotation transfer within Pantools. The following questions will be addressed: Does annotation transfer within a pangenome work across the tree of life and between different genome assembly versions, different strains or accession and even different species? Is the transferred annotation of high quality, not only for single copy homologs, but also for gene families with many similar genes and multiple copy homologs? How complete is the transferred annotation? Is the algorithm computationally feasible?

2 Methods

In this section, the algorithm designed for annotation transfer within Pantools will be discussed. Also, the implementation, test data and validation strategies will be explained.

2.1 Algorithm design

Pantools uses a graph database to store pangenomic data instead of a linear representation, so a new strategy is necessary to implement annotation transfer. In Pantools, similar genomic regions share 'nucleotide' nodes, so locating where an annotation should be transferred to in another genome is straightforward. After an appropriate region has been found, annotation transfer is a matter of mapping where exactly the new annotation should be, and whether or not this gene is still valid in this place. Overall, the algorithm works in five steps: finding similar regions, mapping coordinates, transferring coordinates, validating and resolving double transfers. These steps will be described in detail below and an overview can be seen in Figure 1.

2.1.1 Region finding

The first step in the annotation transfer algorithm is finding similar regions (Figure 1A). For each reference gene that is to be transferred, the sequence of nucleotide nodes that the gene spans is found by its genomic position. For each of these nodes, all in- and outgoing edges to other nucleotide nodes are taken. As edges store the information of which genomic regions go through the nucleotide node, it is now known which other regions share nucleotide nodes with our reference feature's region. Regions that only share less than 5% of our reference nodes are excluded. Next, so called 'anchors' will be made for each of the regions that share nodes. This means that if a region shares more than one consecutive node with the reference, these nodes will be put together in an anchor. Next, 'bridges' are made between anchors, which is done by comparing the sequence length of the unshared regions. If the unshared region is at least 75% and no more than 150% of the reference region's length, and the bridge spans a maximum of 500 bp, a bridge is made. This bridging will be done in several rounds, each round trying to bridge another gap between anchors. Finally, if the length of the matching region is at least 90% of the original feature's sequence length, this region will be reported as a match. There can be zero, one or multiple matches per region. This means that if a gene is matched twice or more, and all following steps will be passed, an annotation is transferred multiple times.

2.1.2 Mapping coordinates

The second step in the annotation transfer algorithm is mapping coordinates for each similar region (Figure 1B). For each matching region and its reference, the nucleotide sequence of both regions is extracted, and an alignment is made with MAFFT on default settings (Katoh & Standley, 2013). Now, each nucleotide coordinate of the reference is associated to the one in the match (except if there is a gap within the alignment). From this alignment, an identity score is also calculated.

2.1.3 Transferring coordinates

The third step in the annotation transfer algorithm is the transferring of begin and end coordinates of the reference features (Figure 1C). For each gene, all child features (mRNA, CDS and exon) are taken from the pangenome. For each matching region, the start and end location of all these features are inferred from the coordinate mapping. If the coordinates of a feature cannot be transferred because a gap in the mapping spans over the begin or end location of the feature, the match is excluded. The type, name and other attributes are copied from the original annotation and a new feature ID is created. The strand is inferred from the reported match direction of the region finding step.

2.1.4 Validating

The fourth step in the annotation transfer algorithm is validating that a transferred gene is still valid in its new place (Figure 1D). This validation will be done per transcript, so it is possible that some transcripts of a gene pass, while another transcript doesn't. The validation rules are: a minimum nucleotide identity of 90%, a nucleotide length of at least 90% of the reference, a minimum protein identity of 90%, a protein length of at least 90% of the reference, a valid start codon (either ATG, or the same codon as the reference to include alternative start codons), a valid stop codon (TAA, TGA or TAG, or the same codon as the reference to include alternative stop codons), a reading frame that is divisible by 3, no in-frame stop codons in exons and finally, a valid splice site in introns (either GT/AG

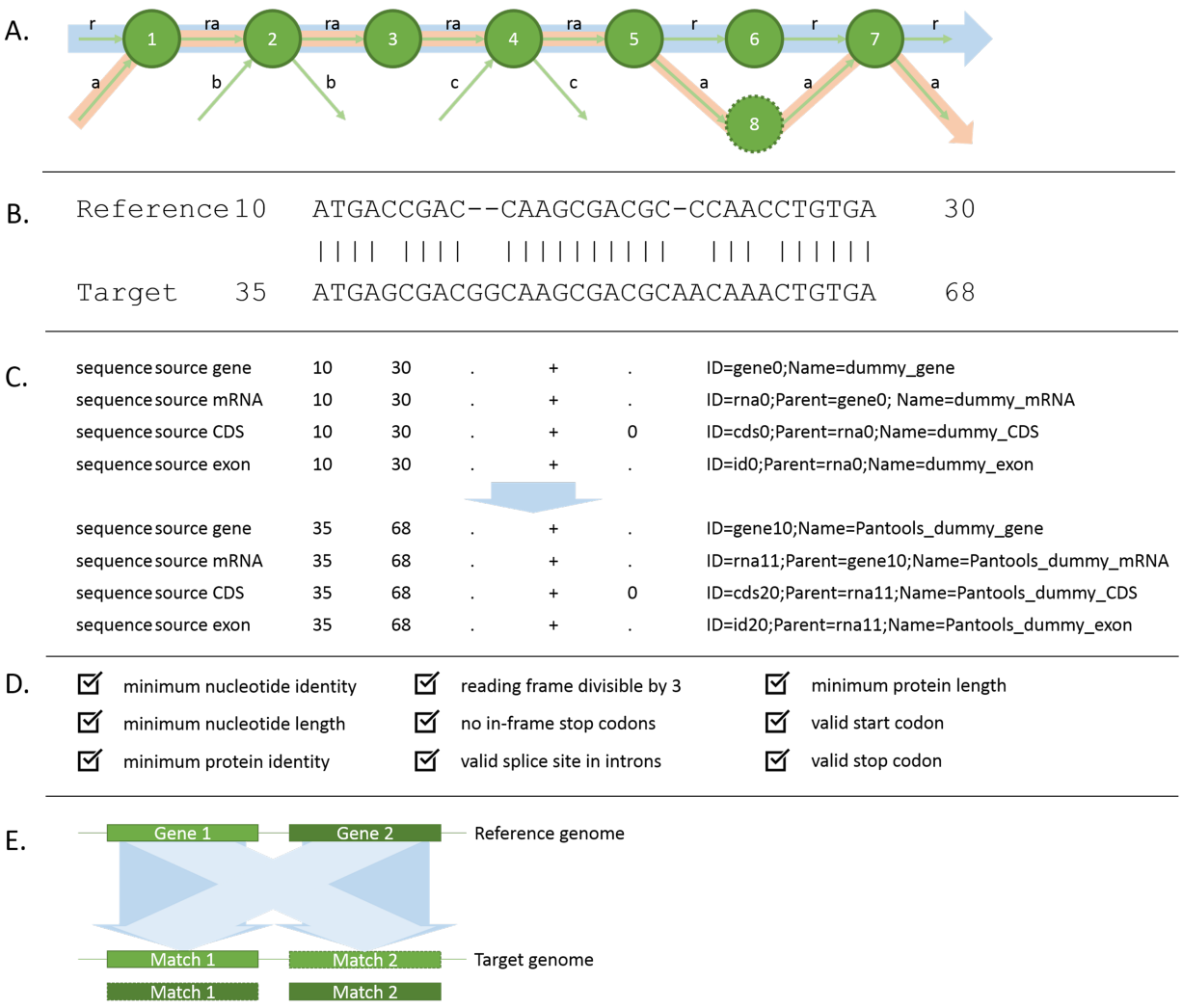

Fig. 1. An overview of the algorithm design. **A. Region finding.** The green circles represent graph nucleotide nodes (which contain nucleotide sequences), the arrows represent edges between these nodes (which contain sequence paths). The letter 'r' indicates the path of the reference region, the letters 'a', 'b' and 'c' indicate other sequences that share a part of the nucleotide nodes. The big blue arrow shows the reference path, and the orange arrow shows the path of sequence 'a'. In this example, it can be seen that region 'a' shares all nodes except node 6 with the reference. Region 'b' and 'c' share only one node each with reference 'a'. Four anchors are made, two of which for sequence 'a': one containing node 1 through 5 and one containing node 7. The other two will contain node 2 and node 4 for region 'b' and 'c' respectively. A bridge will be attempted to made between the two regions of 'a' by comparing node 6 and 8. No bridges can be made for regions 'b' and 'c', as they both only have a single anchor. If the bridge for region 'a' is successful, this will be reported as a match. Region 'b' and 'c' are too short and will be excluded. **B. Mapping coordinates.** A MAFFT alignment is made between the reference sequence and the match sequence and coordinates are mapped. In this example alignment, nucleotide coordinate 10 from the reference is mapped to nucleotide 35 from the match sequence, and so on. When there is a mismatch in the alignment like between coordinate 14 in the reference and coordinate 39 in the match, those coordinates are still mapped. However, when there is a gap in the alignment, the nucleotides spanning this gap such as coordinate 44 and 45 in the match, are not mapped to another nucleotide. This way, it could happen that the begin or end coordinate of a reference gene does not occur in the coordinate mapping. **C. Transferring coordinates.** All info about a gene and all its child features is taken from the pangenome. Then, the begin and end locations of those features are transferred using the coordinate mapping from the previous step. The other gff fields are created as such: sequence name is transferred using the match location, source of annotation will be Pantools, feature type will be copied from the reference, strand is transferred using the match orientation, score and phase are copied from the reference, a new feature id is made and the correct parent is added, and the rest of the attributes are copied from the reference using the prefix "pantools_". Matches for which one or more child features fails to be transferred are excluded. **D. Validating.** For each gene that is transferred, 9 validation steps need to be passed. If one or more of these fails, the transfer is excluded. **E. resolving double transfers.** For each validated gene, it is checked whether or not there are more than 1 transfers to this location. If so, these double transfers need to be resolved. In this example, Gene 1 and Gene 2 are tandemly repeated in the reference genome, and when trying to transfer these genes, both references match to both target regions in the target genome as they all share sequences. However, Gene 1 might have a slightly higher identity to Match 1 and Gene 2 to Match 2. By picking the transfer with the highest identity, double transfers are resolved.

or the same nucleotides as the reference to include alternative splice sites). If one or more of these fields does not pass, this transcript will be excluded. By default, Pantools' annotation transfer will be strict about these rules. However, it is possible to give Pantools an argument to make it non-strict. Then, transcripts that fail validation will be added with the warning label 'needs_curation'.

2.1.5 Resolving double transfers

The final step in the annotation transfer algorithm is the resolving of double transfers (Figure 1E). If a gene is present in multiple copies in a genome or if two genes are very similar, it might occur that the same matching region is found twice. When this happens, the choice of which reference gene is transferred to this location needs to be made. For each region to which multiple transfers are made, the match with the highest nucleotide identity is picked to be the final transfer. If there are ties in the identity, the first transfer that happened is picked. Although this seems arbitrary, identities of two transfers will only be exactly the same if there are exactly the same amount of differences between them as well. This likely means that the two reference genes are exactly the same, and then, only the naming of the genes differs. As gene naming is universally quite divergent and unstandardized, this is a problem that will need to be dealt with on a larger scale.

2.2 Implementation

Annotation transfer was implemented in the development branch "NickyAnnotationTransfer" of Pantools version 1.3, which is written in Java version 8.0 and uses Neo4j version 3.5.3, including its query language Cypher. The annotation transfer functionality is implemented in three classes: AnnotationTransfer, which implements all functionalities, Match, which stores all values related to a match from the region finding step of the algorithm, and Transfer, which stores all values related to a transfer from after the transfer step of the algorithm. The Pantools development repository is available from https://git.wur.nl/Bif/pangenomics/pantools for people with access.

2.3 Test data and validation

Four aspects of annotation transfer will be tested: performance on transferring single-copy homologs, behaviour when transferring multiple-copy homologs, completeness of the transferred annotation and computational feasibility of the algorithm.

2.3.1 Single-copy homolog transfer

The pangenomes used to test performance on single-copy homologs will contain two genomes, one that will serve as reference genome and one that will serve as target genome (see Table 1 and Table S1-3), which vary on two aspects. Firstly, different organisms will be used to test whether annotation transfer within Pantools works across the entire tree of life. These organisms are *E. coli*, *S. cerevisiae*, *D. melanogaster* and *A. thaliana*, thus including bacteria, fungi, animals and plants. Secondly, different amounts of relatedness between the genomes will be used to get an indication of the amount of relatedness that is necessary for annotation transfer to be a viable strategy. The relations between the genomes will be assembly versions, strains/accessions and different species. Therefore, these tests will cover the three use cases for annotation transfer: transferring between genome assembly versions, transferring between related strains or accessions and transferring between related species. To test the quality of annotation transfer, annotations will be added for both genomes in the pangenome, which allows validation to be done between the transferred annotations and the actual annotations of the target genome. However, before the quality of annotation transfer can be tested, an important question needs to be answered: how do we know which annotation from the reference genome corresponds to which annotation in the other genome? In *E. coli* and *S. cerevisiae*, gene names are quite standardized, which gives a clue, but for *D. melanogaster* and *A. thaliana*, this is certainly not the case. Therefore, we decided to use homology grouping as validation, which conveniently comes as built in functionality in Pantools (Sheikhizadeh *et al*., 2018).

One difficulty with using homology groups as validation is that some genes are part of a gene family with multiple very similar genes and some genes are copied multiple times in the genome, both of which would lead to big homology groups in which it is not clear which gene from the reference genome corresponds to what other gene in the target genome. For annotation transfer quality testing, it would be optimal if every gene from the reference genome was in a homology group with either no other genes, indicating that this gene is not present in the other genome, or with a single gene from the target genome, so it is clear where the annotation transfer should happen to. However, when grouping settings are too stringent, genes might end up alone in a homology group instead of with their orthologous gene, as there might be some differences between the them. On the other hand, if grouping settings are too relaxed, genes might end up in larger gene families instead of only with their one homologous gene. To estimate optimal homology grouping settings in this trade-off, homology grouping was done using the 8 precooked settings for various levels of stringency provided by Sheikhizadeh *et al*. (2018) (see Supplementary chapter 2: Pantools homology group validation). This was done all five test pangenomes containing two genomes used for performance testing on single-copy homologs. For each setting in every pangenome containing two genomes, the number of validatable genes was counted and plotted (Figure S1, S2, S3, S4, S5). From these results, it was concluded that the d1 settings (the most stringent settings) would be best to use for validation, as this setting resulted in the highest amount of validatable genes in each of the pangenomes.

Validation of the transferred annotations will be done by comparing them to the actual annotation of the target genome. This comparison will be done on two aspects: the begin and end location of a gene and the begin and end location of all CDSs of a gene. Precision is defined as the number of single-copy homologs that were transferred correctly over the total number of genes transferred. Recall is defined as the number of single-copy homologs that were transferred correctly over the total number of single-copy homologs that should have been able to be transferred.

2.3.2 Multiple-copy homolog transfer

Of course, genes in a gene family with very similar genes or genes that are present multiple times in the genome will still end up in larger homology groups. A few of these cases will be tested separately from the single-copy homologs on a small scale using manual validation. This will be done using two relevant examples of cases in which multiple-copy homologs might be difficult to transfer between genomes. Firstly, when a gene is part of family in which many similar genes occur, it might be difficult to know where this gene should be transferred to, as all locations where genes from that family occur could be viable targets. Secondly, when a gene is duplicated multiple times in both genomes, all copies of this gene will be viable targets for transfer. For both of these examples, double transfers could occur, meaning that one location in the target genome is annotated multiple times. As can be seen in Fig1E, Pantools resolves these double transfers by picking the match with the highest identity to the target location. To demonstrate this behaviour, the examples that will be used are the very conserved auxin response factor (ARF) gene family in *A. thaliana* (Finet *et al*., 2012) and the tandemly repeated gene CUP1 in *S. cerevisiae* (Zhao *et al*., 2014).

First, ARF genes were tested using the pangenome containing two *A. thaliana* accessions. By looking at all homology groups containing ARF genes (see Figure S6 and Supplementary chapter 3: Multiple-copy

Table 1. Pangenomes used as test data. In the pangenome column, the general scope of the pangenome is given. Then, the reference genome is the genome whose annotation will be used to transfer onto the other genomes. For the A. thaliana assembly versions, no assembly accession code is given, because TAIR8 is outdated. Then, the number of scaffolds, the genome size and the number of genes are given for the reference genome, information taken from NCBI's Genome database (Sayers *et al*., 2019). Finally, the number of genomes used in each pangenome is given. For some groups, multiple pangenomes of different sizes are given as they are used to test runtime.

Pangenome	Reference genome	Scaffolds	Genome size (Mb)	Genes	Number of genomes
E. coli **strains**	K-12 substr. MG1655 (GCA_000005845.2)	1	4.6	~4600	2
S. cerevisiae **strains**	S288C (GCA_000146045.2)	17	12.2	~6400	2, 10, 50, 94
Drosophila **species**	*D. melanogaster* Iso-1 (GCA_000001215.4)	1870	143.7	~17700	2
A. thaliana **accessions**	Col-0 (GCA_000001735.2)	7	119.7	~38300	2
A. thaliana **assembly versions**	Col-0 (TAIR8)	7	119.7	~33300	2

homolog transfer), it was decided to test on 4 specific ARF genes that were both in a homology group with one other gene, allowing validation, but also very similar to genes in another homology group, causing ambiguity in where a transfer should happen to. The first ambiguous case are the genes ARFB1B and ARFB1C, who are similar to their corresponding gene in the second genome, but also to each other. The second ambiguous case are the genes ARF1, ARFA1B, ARF1A1C, ARFA1D and ARFA1E. These genes are all similar to each other and their corresponding genes in the second genome.

Secondly, to show how Pantools deals with genes for which many matches will be found, CUP1 was transferred between *S. cerevisiae* strains with a varying amount of copies of this gene. In Strope *et al*. (2015) supplementary table S15, for each of the 100 yeasts strains used there, the copy number of the CUP1 gene was estimated using RNA-seq coverage data. Reference strain S288, which contains two CUP1 copies, was used as reference genome to transfer annotations to target strains YJM1304, YJM1133, YJM1383, YJM993, YJM1527, YJM189, YJM1381, YJM1574, YJM1326 and YJM1549. The number of CUP1 genes present in these strains is 1, 2, 3, 4, 5, 6, 9, 12, 15 and 18, respectively.

2.3.3 Transferred annotation completeness

As these measures only consider the genes that should have been transferred and not the genes that the target genome should have in total, another analysis was done to show the completeness of a transferred annotation. All genes from the reference genome in the pangenome containing two *E. coli* strains will be transferred to the target genome. The feature density of the transferred annotation will be plotted for the entire genome and compared to the feature density of the actual annotation of this genome. For the regions in which there might not be enough features annotated, it will be checked whether this is because of a flaw in the annotation transfer or because this region is not conserved between the two genomes by making a whole genome alignment using Mauve on default settings (Darling *et al*., 2004).

2.3.4 Runtime and peak memory

To test whether or not Pantools' annotation transfer is computationally feasible, three tests will be done, as there are three factors that influence Pantools' annotation transfer runtime and peak memory: the number of genes, genome size and pangenome size. Firstly, it is important that Pantools can deal with a large number of genes, as plants for example have tens of thousands of them. Secondly, it is important that Pantools can deal with large genome sizes too, as genomes can be very large in both the plant and animal kingdoms. Finally, the number of genomes in a pangenome are an important factor as well, as comparative genomics will likely use large pangenomes.

To show impact of the number of genes to be transferred on runtime and peak memory, an increasing amount of genes (100, 200, 300 and 400) was transferred in the *S. cerevisiae* pangenome containing 2 genomes. To test how much genome size matters for runtime and peak memory, annotation transfer was done in 4 pangenomes containing 2 genomes of organisms with increasing genome size: *E. coli*, *S. cerevisiae*, *A. thaliana* and *Drosophila*, of which genome sizes are 4.6 Mb, 12.2 Mb, 119.7 Mb and 143.7 Mb, respectively. For each pangenome, 100 genes from the reference genome were transferred to the other genome in the pangenome. To test how the number of genomes in the pangenome impacts the runtime, annotation transfer was done from one reference genome to an increasing amount of target genomes (1, 9, 49 and 93) within four *S. cerevisiae* pangenomes. Currently, region finding is the most time-consuming step in the algorithm, and this step is done for all genomes at the same time. Therefore, it is possible that the number of genomes in a pangenome doesn't impact runtime quadratically. For each pangenome, 100 annotations from the reference genome were transferred to all other genomes in the pangenome.

3 Results and discussion

Annotation transfer was implemented as an easy to use and transparent functionality in Pantools. It is now possible, when one or multiple new genomes are sequenced for comparative genomics, to annotate them using annotation transfer within a pangenome. The user is informed about the results via multiple log files that document each step in the algorithm. It is now also possible to view or remove annotations, both imported and transferred. When it becomes desirable to use the transferred annotation for analyses outside of a pangenome, it is possible to export them to a standard gff3 file format. Overall, the newly implemented functionalities in Pantools are listed below.

1. *Transfer annotations*, given a reference genome number and one or multiple target genome numbers, transfers genes from the former to the latter. The optional strict/non-strict argument to also add transfers that have failed validation but have passed all other steps. The label 'needs_curation' will be added to these transfers.
2. *View annotations*, given a genome number, shows the annotation ID, the number of features and the source of annotation (from a file or from an annotation transfer) for each annotation of the given genome.
3. *Remove annotation*, given an annotation ID (which can be obtained with *view annotations*), removes the annotation node and all annotation features belonging to this annotation ID.
4. *Remove transferred annotations*, given a genome number, removes all transferred annotations within this genome. When genome number 0 is provided (which is never an actual genome), all transferred annotations in the entire pangenome are removed.
5. *Export gff3*, given a genome number, exports a gff3 file with all annotation features of this genome.

Table 2. The percentage of total genes transferred is shown in the second column, and the percentage of genes that pass in each step of the annotation transfer algorithm for each of the five tested pangenomes are shown in the third, fifth, sixth, seventh and eighth columns. In the fourth column, the percentage of extra matches in the total amount of matches is shown.

Pangenome	**Genes transferred**	Region finding	Extra matches	Coordinate mapping	Transferring	Validating	Resolving double transfers
E. coli **strains**	96.5%	98.9%	1.8%	100.0%	99.5%	97.6%	99.2%
S. cerevisiae **strains**	99.5%	99.8%	11.3%	100.0%	93.3%	99.1%	95.9%
Drosophila **species**	65.4%	85.4%	16.0%	100.0%	97.6%	69.0%	99.9%
A. thaliana **accessions**	91.5%	94.2%	9.8%	100.0%	93.9%	94.2%	97.9%
A. thaliana **assembly versions**	95.2%	95.4%	2.9%	100.0%	98.6%	99.2%	99.1%

The next four sections will show results of the quality and feasibility tests on Pantools' annotation transfer and show that this functionality will be a valuable asset in all fields that use comparative genomics.

3.1 Single-copy homolog transfer

In this section, it is shown that most single-copy homologous genes can be transferred between genomes. Also, it is shown that the quality of transfer is very good transferring CDSs but not so good when transferring UTRs.

Firstly, it is shown that for annotation transfer between assembly versions and strains/accessions, most single copy homologs can be transferred. As we're only transferring single copy homologous genes, all genes should be transferred to the target genome exactly once, and no additional extra matches should be found in the region finding step of the algorithm. The overall percentage of genes that could be transferred in each pangenome is shown in the second column of Table 2. As can be seen, for all pangenomes except *Drosophila*, more than 90% of genes can be transferred. This shows that transferring annotations is a very feasible approach between genome assembly versions and strains/accessions. Although most genes are transferred in *Drosophila* as well, more tests need to be done to get a better idea of how well annotation transfer will work between species. A start was made with this analysis, but it was not finished due to time restrictions (see Supplementary chapter 4: Minimum genome relatedness).

To demonstrate where improvements can still be made in the algorithm, the percentage of matches that passes each step in the algorithm is also plotted in Table 2 and Sankey flow diagrams of both the best (*S. cerevisiae*) and worst (*Drosophila*) transfers is shown in Figure 2. For the other three pangenomes, the flow diagrams are shown in figures S14-16. In the region finding step, matching regions are found for practically all genes in *S.*

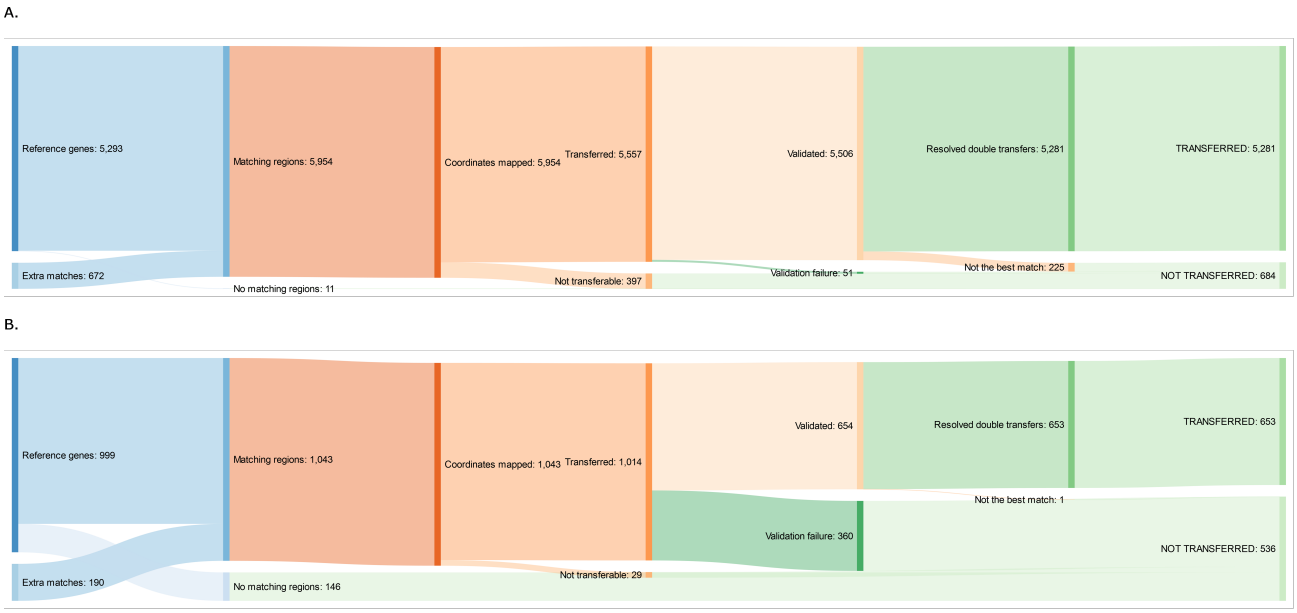

Fig. 2. A. Single-copy homolog transfer results for *S. cerevisiae*. In the region finding step, for 5281 out of 5293 genes, matching regions are found (99.8%). The other 11 genes had no matching regions. There are also 672 additional matches from genes that had more than one matching region. For all 5954 matching regions, nucleotide coordinates were mapped. In the transfer step, 5557 of the 5954 matches (93.3%) can be transferred successfully. This means that the other 397 matching regions match only part of the gene region and missed either the begin or end location in the coordinate mapping index. When validating the 5557 transferred genes, 5506 (99.1%) pass. For the resolving of double transfers in one region, 5281 out of 5506 (95.9%) validated transfers are the best match in that location. Overall, of all 5293 genes that the algorithm started with, 27 were not transferred (meaning that most dropouts were extra matches), and 4 genes were transferred twice. **B. Single-copy homolog transfer results for *Drosophila* species.** In the region finding step, for 853 out of 999 genes, matching regions are found (85.4%). The other 146 genes had no matching regions. There are also 190 additional matches from genes that had more than one matching region. For all 1043 matching regions, nucleotide coordinates were mapped. In the transfer step, 1014 of the 1043 matches (97.2%) can be transferred successfully. This means that the other 29 matching regions match only part of the gene region and missed either the begin or end location in the coordinate mapping index. When validating the 1014 transferred genes, 654 (64.5%) pass. For the resolving of double transfers in one region, 653 out of 654 (99.8%) validated transfers are the best match in that location. Overall, of all 999 genes that the algorithm started with, 346 were not transferred (meaning that most dropouts were extra matches), and 0 genes were transferred twice.

cerevisiae, but 15% of genes is already not found in *Drosophila*. Although this makes sense, as there is more variation between the two genomes in the *Drosophila* pangenome as they are of different species, it might be something that can be improved upon in the future. As can also be seen, there are extra matches in the region finding step for each pangenome. As each gene in this test should be transferred exactly once, those extra matches shouldn't be there. What is causing these extra matches in *E. coli* and *S. cerevisiae* is a bug which reports certain matches twice with a slightly varying start or end location. Upon further inspection of the extra matches in *Drosophila* and both *A. thaliana* transfers, it was discovered that these were partly caused by the same bug as in *E. coli* and *S. cerevisiae*, but also partly caused by a bug in the writing of the log file, which reports each transcript of a gene as an extra match. These bugs should be fixed, but do not impact the algorithm, as double matches are filtered out during the last step of the algorithm. For coordinate mapping, there are currently no elimination rules, so all matches pass here. When checking a few low-identity matches, it became apparent that most of these matches were partial, spanning only the last half of the reference gene. The reason why these partial matches were still long enough to be reported by Pantools as a matching region is because the match sequence continues further before or after gene. These low identity matches are not a big problem, as they will be filtered out in the next step, but this does indicate that in the future, another filtering step could already be implemented here which excludes matches with a very low coverage. For transferring coordinates, only *S. cerevisiae* and *A. thaliana* show a slightly lower percentage of genes that pass. The reason for this is that they both have some extra matches which are only partial in the region finding step, which are all eliminated in the transferring step. Most matches pass the transferring step in the *Drosophila* pangenome as these al turned out to be transcripts instead of extra matches. In the validation step, only *Drosophila* shows a very low number of genes that pass. When looking at which of the 9 validation rules is causing this, it becomes apparent that about 15% of all matches fail because the minimum match identity is lower than 90%. Therefore, it seems that this rule is too stringent when transferring between different species, and more testing needs to be done to get a better threshold. Also, another 10% of matches fail because of an invalid stop codon, suggesting that the algorithm could benefit greatly from having an auto-correct implementation, which looks up- and downstream for a certain amount of base pairs to find a valid stop codon. This could also lead to an even higher percentage of matches that pass in the other four pangenomes. In the resolving of double transfers, most genes pass in all pangenomes. A few less pass in the *S. cerevisiae* and *A. thaliana* transfers because of the bug that was reporting the same match twice. Overall, these results show that besides a few small improvements that can be made, the annotation transfer algorithm works very well in a variety of species and different amounts of relatedness.

Secondly, it is shown that Pantools' annotation transfer has good precision and recall when transferring CDSs, but a low precision and recall when transferring UTRs. For all 5 pangenomes, of the annotations that were transferred, the results were compared to the actual annotation of the target genome to get precision and recall measures. Two measures were used for validation. First, a transferred gene was counted as transferred correctly if the begin and end location and the strand of this gene are correct. However, according to this measure, the intron and exon structure of the gene could be completely wrong and it would still be counted as transferred correctly. Therefore, secondly, another validation was done where a transferred gene was counted as transferred correctly if all CDSs of all transcripts were correct. This means that if there was a transcript missing or a transcript extra, this genes is also not counted as transferred correctly. Precision and recall for both of these measures for all 5 pangenomes can be seen in Table 3.

As can be seen, for the validation of gene begin and end location, the precision and recall vary a lot. Precision and recall are extremely good in both *S. cerevisiae* and *A. thaliana* assembly versions, reasonable in *E. coli*, and terrible in *A. thaliana* and *Drosophila*. Upon further investigation, it turned out that there were no UTRs annotated in either *E. coli* or *S. cerevisiae*, while they were annotated in *A. thaliana* and *Drosophila*. Therefore, it is likely that transferring UTRs, which have been shown to not be very conserved (Lin & Li, 2011), is not possible without additional RNA-seq data, and that if UTRs were annotated in *E. coli* and *S. cerevisiae* as well, the precision and recall of their transfers would be worse as well. The precision and recall of CDSs of transferred genes is much better in *A. thaliana* and *Drosophila* than those of the gene validation, and they are similar in the other three pangenomes. Therefore, it is concluded that annotation transfer works reasonably well for CDSs, but not so well for UTRs.

Table 3. Precision and recall of annotation transfer of single-copy homologs in a variety of pangenomes. Two different validation strategies are shown, the first one being gene validation and the second one being CDSs validation.

Pangenome	Gene validation		CDS validation	
	Precision	Recall	Precision	Recall
E. coli **strains**	0.661	0.641	0.657	0.637
S. cerevisiae **strains**	0.997	0.995	0.996	0.994
Drosophila **species**	0.007	0.005	0.812	0.531
A. thaliana **accessions**	0.150	0.138	0.864	0.779
A. thaliana **assembly versions**	0.939	0.894	0.934	0.888

Even though precision and recall are better when validating using CDSs rather than UTRs, improvements are still necessary. Firstly, in *E. coli*, about a third of all genes is transferred to the correct location in the target genome, but to the wrong strand. This is happening because there is a large inversion in the target genome compared to the reference genome (see Figure 3A). Therefore, finding and fixing this bug will likely raise precision and recall to close to 1 for *E. coli*. Secondly, to improve recall in *Drosophila*, more testing should be done to optimize the 9 gene validation rules in the validation step of the algorithm. Thirdly, to improve precision overall, RNA-seq data could be integrated to improve both CDS precision even further, but also to improve UTR precision. Fourthly, to improve recall overall, an auto-correct could be implemented to save genes that have an invalid start codon, stop codon or splice site after transfer. When these improvements are implemented in Pantools' annotation transfer, transferred annotations will be of very high quality.

Overall, the transfer of single-copy homologs is going very well in similar genomes such as assembly versions and strains/accessions, and reasonably well in less similar genomes of different species. Therefore, these results show that Pantools can reliably transfer single-copy homologs to another genome between assembly versions and strains, but that more research is necessary on the amount of relatedness that is necessary between genomes for annotation transfer to be a viable strategy. As annotation transfer will most likely predominantly be used to transfer annotations between assembly versions and strains for comparative genomics, it is already very useful a functionality within Pantools.

3.2 Multiple-copy homolog transfer

In this section, it is shown that Pantools' annotation transfer can not only deal with transferring single-copy homologous genes, but also multiple-copy homologs.

Firstly, four ARF genes which are part of a large gene family in *A. thaliana* with many similar genes were transferred between two genomes. When transferring these genes, not only were all four transferred to the correct region, but the region finding step in the transfer algorithm

found only the correct region. The reason why Pantools has no trouble transferring these genes correctly, despite the presence of many other homologous genes, is because it looks at similarity on the nucleotide level, not on the protein level. Even though ARF's are very conserved genes because of their important function in plant development, this conservation is mostly on the protein level. Therefore, other homologous genes can be very different on the nucleotide level, which causes no problems for Pantools' annotation transfer. Besides this, the region finding step considers the entire gene length including introns, which are not as conserved as exons and aren't considered in homology grouping. This result shows that transferring genes in families with many similar and very conserved genes will not be a problem in Pantools.

Secondly, the CUP1 genes in *S. cerevisiae*, which are tandemly repeated in varying amounts in different strains, were transferred to several strains. These tandemly repeated genes are all exactly the same, so unlike the ARF gene family, this will cause double matches to be found in the region finding step. In the genomes of the 10 strains the CUP1 genes were transferred to, there should be 1, 2, 3, 4, 5, 6, 9, 12, 15 and 18 copies of this gene present, respectively. However, when both CUP1 genes from the reference are transferred to the 10 other genomes, exactly one matching region is found in each genome, twice for each reference CUP1 gene. From these results, it was concluded that the region finding algorithm has trouble finding tandemly repeated sequences. Despite this problem however, the one matching region in each genome was in fact found twice, once for each reference CUP1 gene. If it wasn't for Pantools resolving these double transfers, two annotation would en up in the same location in each of the genomes. However, because Pantools resolves these double transfers, the first CUP1 gene is transferred 10 times and the second CUP1 gene zero times, because Pantools transfers either the match with the highest identity or, if the identities are the same, the first match. Because of this, there are no double annotations in the target genomes. These results show that Pantools' annotation transfer algorithm can also deal with repeated genes.

All in all, Pantools is able to deal with multiple-copy homologs very well. However, this analysis did bring to light the fact that improvements should be made in the finding matching regions algorithm to deal with tandemly repeated regions.

3.3 Completeness of transferred annotation

In this section, it is shown that even with pairwise annotation transfer within a pangenome, a largely complete annotation can already be produced for a newly sequenced genome.

All annotations from the reference genome in the *E. coli* pangenome were transferred to the target genome. To see how complete the transferred annotation is, the feature density of the transferred annotation was compared to the feature density of the actual annotation of the target genome. The results of this can be seen in Figure 3B. As can be seen, the feature density of both annotation is comparable, although that of the transferred annotation is somewhat lower. However, there are some gaps were there are no annotated features in the transferred annotation. To see whether or not this was because these regions are not conserved between the two genomes, a whole genome alignment was made (see Figure 3A). As can be seen, the gaps in the feature density correspond exactly to the regions where there is no conservation between the two genomes. These results show that the transferred annotation is as complete as it can get using annotation transfer with a single reference genome.

Because of this result, a question that needs to be raised is: can the annotation of a genome ever be complete by using annotation transfer solely, as there is no way to annotate genes that are not present in the reference genome? The solution to this problem is twofold. Firstly, it was shown by Tettelin *et al.* (2005) that as the number of genomes in a pangenome increases, the number of different genes present in the pangenome also increases. Therefore, if most genes in a population are present in the pangenome, using multiple references for annotation transfer would make the transferred annotation more complete. This is valuable

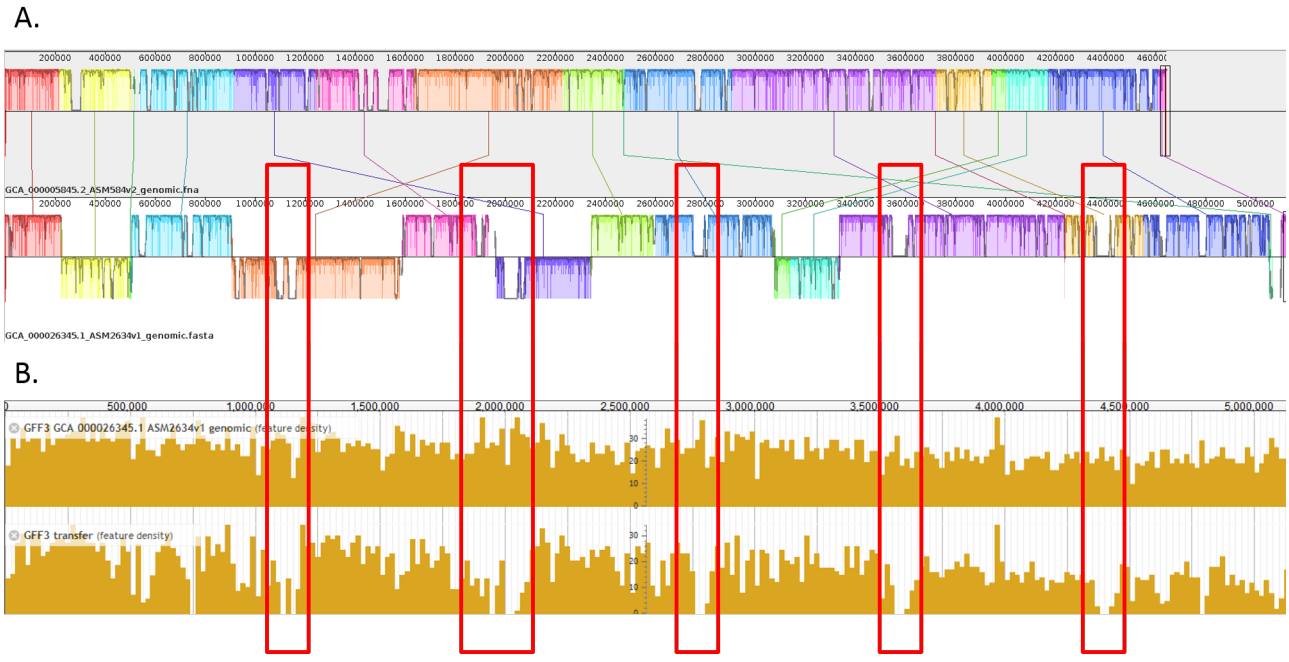

Fig. 3. A. Whole genome alignment of two *E. coli* strains. Conserved blocks are indicated by color, and the amount of conservation is indicated by the height of the block. On the top, the reference genome is shown and on the bottom, the target genome is shown. **B. Feature density plot of actual and transferred annotations. On the top, the actual feature density of the target genome is shown and on the bottom, the transferred annotation feature density is shown. As indicated by the red squares, regions where there are no annotations in the transferred annotation are also regions where there is no conservation between the two genomes.**

functionality that should be implemented in a future version of annotation transfer within Pantools. However, one downside of annotation transfer will always be that it is impossible to find novel genes, a certain amount of which are still found with each newly sequenced genome (Tettelin *et al*., 2005). Therefore, secondly, in order for Pantools to annotate these novel genes as well, RNA-seq data should be integrated as well.

Despite the fact that currently, only single reference annotation is implemented in Pantools, it already provides a largely complete annotation and it has a lot of potential to produce an even more complete annotation.

3.4 Runtime and peak memory

In this section, it is shown that Pantools' annotation transfer algorithm is computationally feasible. Three factors were tested: the number of genes being transferred, genome size and pangenome size, the results of which can be seen in Figure 4.

Firstly, it is shown that the algorithm scales well to a large number of genes being transferred. By transferring an increasing amount of genes (100, 200, 300 and 400) while keeping genome size and pangenome size the same with an *S. cerevisiae* pangenome containing two genomes, a relationship between CPU runtime and peak memory could be obtained. These results are shown in Figure 4 A and B. As can be seen, there is a linear relationship between the number of genes being transferred and the runtime ($R^2 = 0.9852$). Extrapolating from this relationship to the amount of genes that, for example, *A. thaliana* has, which is about 33.000, this would result in a CPU time of about 5 hours. There seems to be no relationship between the peak memory and the number of genes being transferred ($R^2 = 0.0009$). Although, the number of genes that is being transferred needs to be tested on a larger scale to get a definitive estimate about runtime and peak memory, these results show that a large amount of genes can be transferred within a reasonable amount of time.

Secondly, it is shown that the algorithm scales extremely well to transferring annotations within a large genome. The relationship between genome size and runtime and peak memory was tested by transferring 100 genes between genomes of increasing size (4.6, 12.2, 119.7 and 143.7 Mb) within a pangenome containing 2 genomes. The results of this experiment are shown in Figure 4 C and D. As can be seen, there is a weak relationship between the runtime and genome size, but only a little variance can be explained by genome size alone ($R^2 = 0.2352$). The same is the case for the relationship between peak memory and genome size ($R^2 = 0.1632$). These results show that there are likely other more important differences between these pangenomes that impact runtime and peak memory. The reason why genome size doesn't have a big impact on runtime and peak memory is because similar genomic regions are already together within the structure of the pangenome. This means that the algorithm only has to look in this region and not in the rest of the genome. I suggest that other impact factors could include gene length, the number of repeated regions within the genomes and the relatedness between the two genomes. Longer genes could be a cause of longer computation time in every step of the algorithm, more repeated regions in the genomes could cause more computation time as more matching regions will be found and less relatedness between the genomes could cause shorter computation time as more often, no matching region are found. Overall, both runtime and peak memory increase only marginally when annotations are transferred between larger genomes. These results suggest that Pantools' annotation transfer can likely deal with even larger genomes such as that of wheat.

Finally, it is shown that more improvements need to be made to the algorithm before annotation transfer becomes feasible within large pangenomes. To test the impact of pangenome size on runtime and peak memory, 100 genes were transferred within an *S. cerevisiae* pangenome from one reference genome to an increasing amount of target genomes (1, 9, 49, 93). The the results of this are shown in Figure 4 E and F. As can be seen, there is a strong positive relationship between both the runtime and pangenome size ($R^2 = 0.9923$) and peak memory and pangenome size ($R^2 = 0.9829$). For only 100 genes, CPU runtime is already over an hour when there are 94 genomes present in the pangenome. Extrapolating from this relationship to the amount of genes that *S. cerevisiae* really has, and transferring those within a pangenome containing 94 genomes, this would result in a CPU time of almost 3 days. With 94 genomes in the pangenome, peak memory becomes four times higher than for the pangenome with 2 genomes. These results show that Pantools' annotation transfer doesn't scale quite satisfactorily with large pangenomes yet for the two reasons. Firstly, although runtime and peak memory are still within a reasonable range for 94 *S. cerevisiae* genomes, the large amount of genes in other organisms will make annotation transfer quite a long process. Secondly, as shown by many big sequencing projects (Li *et al*., 2014; Gilbert *et al*., 2014; Siva, 2008), it is safe to assume that even larger pangenomes will become a reality in the future, which will raise runtime and peak memory even higher. Besides this, there currently is another problem with annotation transfer within large pangenomes. This problem is that in the region finding step of the algorithm, it is currently impossible to find matching regions for only a subset of genomes in a pangenome. As it usually will be the case that only a few new genomes will be added to a large pangenome, a lot of computational time is wasted by finding matching regions in genomes that already have annotation. Overall, more work needs to be done to make annotation transfer scalable to larger pangenomes.

All in all, these results show that Pantools' annotation transfer scales very well to a large amount of genes and large genomes. However, improvements will need to be made before it is also scalable when big pangenomes are used. For these improvements, I have the following four suggestions. Firstly, the region finding step, which is currently implemented in python, should be updated to the newest version which is written in Java. Updating this step will improve runtime significantly because it eliminates the need for an external process to run within Pantools, because Java is generally believed to be faster, and mostly because the algorithm is redesigned to be faster in and of itself. Secondly, another bottleneck in the algorithm is the coordinate mapping step which is currently done using MAFFT. MAFFT could be replaced with Pantools' built-in alignment functionality, which might improve runtime as this also eliminates the need for an external process to run within Pantools. However, it remains to be tested whether replacing MAFFT is actually faster. Thirdly, as genes are independently transferred until the last step, resolving double transfers, multi-threading could be implemented to greatly benefit runtime. Fourthly, it might be possible to speed up the region finding process by not doing region finding for one gene at a time, but for multiple neighbouring genes at a time. However, it needs to be tested whether or not low complexity regions in between those genes might increase runtime again. An even faster option might be to approach the region finding step in a whole-genome alignment way.

When these improvements can be made, it will be more scalable to use Pantools' annotation transfer within large pangenomes as well as within small ones.

4 Conclusion

Comparative genomics is an increasingly relevant field of research as it is now easier and cheaper to sequence genomes than ever, yielding a huge wealth of genomic data. As annotations give biological context to genomic variation found in comparative genomics, it is important that annotation efforts keep up with sequencing efforts. To this end, annotation transfer was implemented as a built in functionality in Pantools. It is able to quickly and cheaply annotate a newly sequenced genome that is related to an already present annotated genome without a lot of manual curation being

necessary. It was shown that most single-copy homologous genes can be transferred between genomes of many different species and between genomes of assembly versions and strains/accessions. However, there were some minor bugs in the algorithm that should be straightened out in order to be able to transfer all genes. Between species, a decent amount of genes was also shown to be transferable, but more testing is needed before this

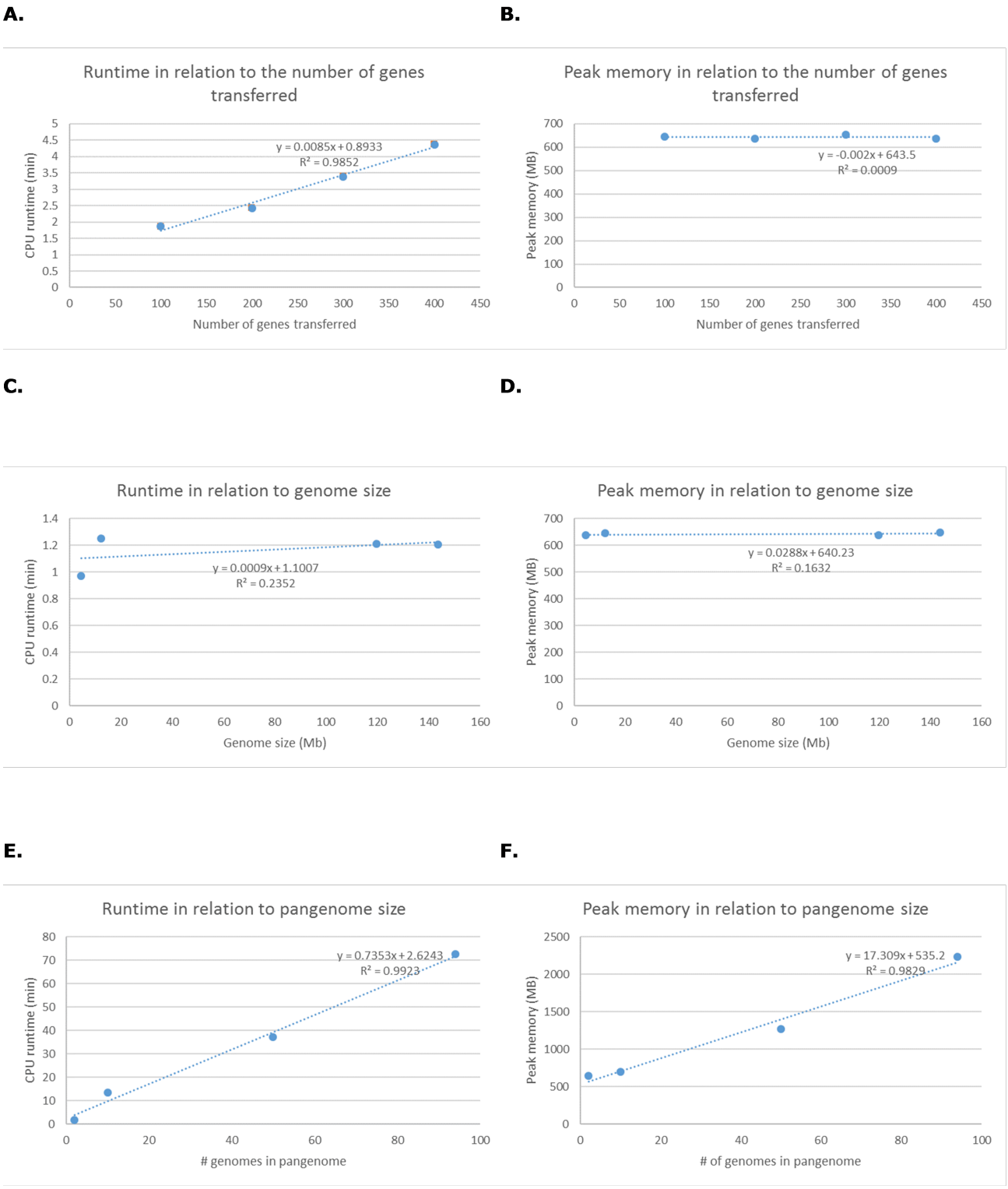

Fig. 4. Annotation transfer runtime and peak memory in relation to three factors: number of genes to be transferred (**A.** and **B.**), genome size (**C.** and **D.**) and pangenome size (**E.** and **F.**). In A., C. and E., the CPU runtime is shown in minutes, and in B., D. and F., the peak memory is given in MB. For all, a linear regression line is shown as well as an R^2 value to show how much variability is explained by the linear regression model.

can be applied with reliable results. Of the genes that were transferred, the quality of transfer of CDSs was very good. However, in order to be able to transfer UTRs successfully also, additional RNA-seq data needs to be integrated. Pantools' annotation transfer can not only transfer single-copy homologous genes, but also multiple-copy homologs. It was shown that annotation transfer within a pangenome is a viable strategy to get a relatively complete annotation for a newly sequence genome. Also, Pantools' annotation transfer algorithm is computationally feasible to transfer many genes in big genomes. However, some improvements should be made before it also scales well to large pangenomes.

This work has three considerable limitations. The first is that due to time restrictions, annotation transfer was only implemented in a pairwise manner, thus not making use of the advantage a pangenome has, which is that there are many reference annotations available. The second limitation is that no benchmarking was done to other annotation transfer tools. Therefore, it is unsure whether or not this new functionality in Pantools is better than already existing tools. The third limitation is that currently, it is only possible to transfer coding genes and no other features. However, Pantools' annotation transfer shows that it is able to do high-quality annotation transfers while using a single reference genome only. Also, additional features could easily be added to the annotation transfer algorithm. Therefore, there is a lot of potential for annotation transfer quality to become even higher when multiple reference annotation transfer is implemented, when RNA-seq data is integrated and when more features such as breeding markers or tRNAs are added to the transfer algorithm as well.

Acknowledgements

Thanks to Eef Jonkheer, Siavash Sheikhizadeh and Leonoor Engeltjes for all their help.

References

The Computational Pan-Genomics Consortium (2016). Computational pan-genomics: status, promises and challenges. *Briefings in bioinformatics*, **19**, 118-135.

Darling, A. C., Mau, B., Blattner, F. R., & Perna, N. T. (2004). Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome research*, **14(7)**, 1394-1403.

Fiddes, I. T., Armstrong, J., Diekhans, M., Nachtweide, S., Kronenberg, Z. N., Underwood, J. G., ... & Haussler D., Stanke M. & Paten B. (2018). Comparative Annotation Toolkit (CAT)-simultaneous clade and personal genome annotation, *Genome research*, **28**, 1029-1038.

Finet, C., Berne-Dedieu, A., Scutt, C. P., & Marlétaz, F. (2012). Evolution of the ARF gene family in land plants: old domains, new tricks. *Molecular Biology and Evolution*, **30(1)**, 45-56.

Fleischmann, R. D., Adams, M. D., White, O., Clayton, R. A., Kirkness, E. F., Kerlavage, A. R., ... & Merrick, J. M. (1995). Whole-genome random sequencing and assembly of Haemophilus influenzae Rd, *Science*, **269(5223)**, 496-512.

Fundel, K., & Zimmer, R. (2006). Gene and protein nomenclature in public databases. *Bmc Bioinformatics*, **7(1)**, 372.

Gilbert, J. A., Jansson J. K., & Knight R. (2014). The Earth Microbiome project: successes and aspirations, *BMC Biology*, **12**, 69.

Hoff, K. J., Lange, S., Lomsadze, A., Borodovsky, M., & Stanke, M. (2015). BRAKER1: unsupervised RNA-Seq-based genome annotation with GeneMark-ET and AUGUSTUS, *Bioinformatics*, **32(5)**, 767-769.

Holt, C., & Yandell, M. (2011). MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects, textitBMC bioinformatics, **12**, 491.

Katoh, K., & Standley, D. M. (2013). MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, **30(4)**, 772-780.

Li, J., Wang, J., & Zeigler, R. S. (2014). The 3,000 rice genomes project: new opportunities and challenges for future rice research, *GigaScience*, **3**, 8.

Lin, Z., & Li, W. H. (2011). Evolution of $5'$ untranslated region length and gene expression reprogramming in yeasts. *Molecular biology and evolution*, **29(1)**, 81-89.

Otto, T. D., Dillon, G. P., Degrave, W. S., & Berriman, M. (2011). RATT: rapid annotation transfer tool, *Nucleic acids research*, **39(9)**, e57.

Sayers, E. W., Agarwala, R., Bolton, E. E., Brister, J. R., Canese, K., Clark, K., ... & Holmes, J. B. (2019). Database resources of the National Center for Biotechnology Information. *Nucleic acids research*, **47(Database issue)**, D23.

Sheikhizadeh, S., Schranz, M. E., Akdel, M., de Ridder, D., & Smit, S. (2016). PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*, **32(17)**, i487-i493.

Sheikhizadeh, S., de Ridder, D., Schranz, M. E., & Smit, S. (2018). Efficient inference of homologs in large eukaryotic pan-proteomes. *BMC bioinformatics*, **19(1)**, 340.

Strope, P. K., Skelly, D. A., Kozmin, S. G., Mahadevan, G., Stone, E. A., Magwene, P. M., ... & McCusker, J. H. (2015). The 100-genomes strains, an S. cerevisiae resource that illuminates its natural phenotypic and genotypic variation and emergence as an opportunistic pathogen. *Genome research*, **25(5)**, 762-774.

Siva, N. (2008). 1000 Genomes project, *Nature Biotechnology*, **26**, 256.

Tettelin, H., Masignani, V., Cieslewicz, M. J., Donati, C., Medini, D., Ward, N. L., ... & DeBoy, R. T. (2005). Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: implications for the microbial "pan-genome". *Proceedings of the National Academy of Sciences*, **102(39)**, 13950-13955.

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., ... & Gocayne, J. D. (2001). The sequence of the human genome. *Science*, **291(5507)**, 1304-1351.

Zhao, Q., Feng, Q., Lu, H., Li, Y., Wang, A., Tian, Q., ... & Huang, X. (2018). Pan-genome analysis highlights the extent of genomic variation in cultivated and wild rice. *Nature genetics*, **50(2)**, 278.

Zhao, Y., Strope, P. K., Kozmin, S. G., McCusker, J. H., Dietrich, F. S., Kokoska, R. J., & Petes, T. D. (2014). Structures of naturally evolved CUP1 tandem arrays in yeast indicate that these arrays are generated by unequal nonhomologous recombination. *G3: Genes, Genomes, Genetics*, **4(11)**, 2259-2269.

Supplementary data

Contents

1. Genomes used in pangenomes

Table S1 - Single-copy homolog testing pangenomes

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	E. coli K 12 substr MG1655 uid225	4958 KB	U00096	Reference
2	E. coli IAI39 uid33411	5084 KB	CU928164	Target

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	S. cerevisiae S288C	12.1571	GCA_000146045.2	Reference
2	S. cerevisiae YJM993	12.4989	GCA_000662435.1	Target

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	D. melanogaster Iso-1	143.726	GCA_000001215.4	Reference
2	D. simulans w501	124.964	GCA_000754195.3	Target

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	A. thaliana Col-0	119.669	GCA_000001735.2	Reference
2	A. thaliana Ler-0	113.15	GCA_000835945.1	Target

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	A. thaliana Col-0 (TAIR8)	119.669	-	Reference
2	A. thaliana Col-0 (TAIR10)	119.669	GCA_000001735.2	Target

Table S2 - Multiple-copy homolog testing pangenomes

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	S. cerevisiae S288C	12.1571	GCA_000146045.2	Reference
2	S. cerevisiae YJM993	12.4989	GCA_000662435.1	Target
3	S. cerevisiae YJM189	12.2614	GCA_000975735.3	Target
4	S. cerevisiae YJM1133	12.1187	GCA_000976695.2	Target
5	S. cerevisiae YJM1304	12.3665	GCA_000977025.1	Target
6	S. cerevisiae YJM1326	12.4065	GCA_000977115.2	Target
7	S. cerevisiae YJM1381	12.4062	GCA_000977355.2	Target
8	S. cerevisiae YJM1383	12.5861	GCA_000977385.1	Target
9	S. cerevisiae YJM1527	12.7146	GCA_000978195.1	Target
10	S. cerevisiae YJM1549	13.2166	GCA_000978225.1	Target
11	S. cerevisiae YJM1574	12.2871	GCA_000978285.1	Target

Number	Organism	Genome size (Mb)	Genbank accession	Function
1	A. thaliana Col-0	119.669	GCA_000001735.2	Reference
2	A. thaliana Ler-0	113.15	GCA_000835945.1	Target

Table S3 - Runtime testing pangenomes

Number	Organism	Genome size (Mb)	Genbank accession	2 Genomes	10 Genomes	50 Genomes	94 Genomes
1	S. cerevisiae S288C	12.1571	GCA_000146045.2	Reference	Reference	Reference	Reference
2	S. cerevisiae YJM993	12.4989	GCA_000662435.1	Target	Target	Target	Target
3	S. cerevisiae YJM195	12.7524	GCA_000975585.1	-	Target	Target	Target
4	S. cerevisiae YJM244	12.8576	GCA_000975615.1	-	Target	Target	Target
5	S. cerevisiae YJM1078	11.8576	GCA_000975645.2	-	Target	Target	Target
6	S. cerevisiae YJM1083	12.4684	GCA_000975675.1	-	Target	Target	Target
7	S. cerevisiae YJM1129	13.0833	GCA_000975705.1	-	Target	Target	Target
8	S. cerevisiae YJM189	12.2614	GCA_000975735.3	-	Target	Target	Target
9	S. cerevisiae YJM193	13.0078	GCA_000975765.1	-	Target	Target	Target
10	S. cerevisiae YJM248	12.1126	GCA_000975795.1	-	Target	Target	Target
11	S. cerevisiae YJM270	12.3653	GCA_000975825.1	-	-	Target	Target
12	S. cerevisiae YJM271	13.421	GCA_000975855.2	-	-	Target	Target
13	S. cerevisiae YJM320	13.4761	GCA_000975885.1	-	-	Target	Target
14	S. cerevisiae YJM326	12.3508	GCA_000975915.1	-	-	Target	Target
15	S. cerevisiae YJM428	12.2726	GCA_000975945.1	-	-	Target	Target
16	S. cerevisiae YJM450	12.3888	GCA_000975975.2	-	-	Target	Target
17	S. cerevisiae YJM451	12.1525	GCA_000976005.1	-	-	Target	Target
18	S. cerevisiae YJM453	12.7106	GCA_000976035.1	-	-	Target	Target
19	S. cerevisiae YJM456	12.2745	GCA_000976065.1	-	-	Target	Target
20	S. cerevisiae YJM470	12.5585	GCA_000976095.1	-	-	Target	Target
21	S. cerevisiae YJM541	12.9666	GCA_000976125.1	-	-	Target	Target
22	S. cerevisiae YJM554	13.0198	GCA_000976155.3	-	-	Target	Target
23	S. cerevisiae YJM555	13.2955	GCA_000976185.1	-	-	Target	Target
24	S. cerevisiae YJM627	12.5811	GCA_000976215.1	-	-	Target	Target

25	S. cerevisiae YJM681	12.3431	GCA_000976245.1	-	-	Target	Target
26	S. cerevisiae YJM682	12.2746	GCA_000976275.3	-	-	Target	Target
27	S. cerevisiae YJM683	12.4278	GCA_000976305.1	-	-	Target	Target
28	S. cerevisiae YJM689	12.19	GCA_000976335.1	-	-	Target	Target
29	S. cerevisiae YJM693	13.1201	GCA_000976365.1	-	-	Target	Target
30	S. cerevisiae YJM969	12.359	GCA_000976395.1	-	-	Target	Target
31	S. cerevisiae YJM972	12.9616	GCA_000976425.1	-	-	Target	Target
32	S. cerevisiae YJM975	12.2218	GCA_000976455.2	-	-	Target	Target
33	S. cerevisiae YJM978	12.4892	GCA_000976485.1	-	-	Target	Target
34	S. cerevisiae YJM981	12.8972	GCA_000976515.1	-	-	Target	Target
35	S. cerevisiae YJM984	13.0626	GCA_000976545.1	-	-	Target	Target
36	S. cerevisiae YJM987	12.9731	GCA_000976575.3	-	-	Target	Target
37	S. cerevisiae YJM990	12.1965	GCA_000976605.2	-	-	Target	Target
38	S. cerevisiae YJM996	12.6958	GCA_000976665.1	-	-	Target	Target
39	S. cerevisiae YJM1133	12.1187	GCA_000976695.2	-	-	Target	Target
40	S. cerevisiae YJM1190	12.9065	GCA_000976725.1	-	-	Target	Target
41	S. cerevisiae YJM1199	12.9065	GCA_000976755.1	-	-	Target	Target
42	S. cerevisiae YJM1202	12.5098	GCA_000976785.1	-	-	Target	Target
43	S. cerevisiae YJM1208	13.0594	GCA_000976815.2	-	-	Target	Target
44	S. cerevisiae YJM1242	13.0594	GCA_000976845.2	-	-	Target	Target
45	S. cerevisiae YJM1244	12.6675	GCA_000976875.2	-	-	Target	Target
46	S. cerevisiae YJM1248	12.5207	GCA_000976905.1	-	-	Target	Target
47	S. cerevisiae YJM1250	12.4871	GCA_000976935.1	-	-	Target	Target
48	S. cerevisiae YJM1252	12.5675	GCA_000976965.2	-	-	Target	Target
49	S. cerevisiae YJM1273	12.5399	GCA_000976995.1	-	-	Target	Target
50	S. cerevisiae YJM1304	12.3665	GCA_000977025.1	-	-	Target	Target

51	S. cerevisiae YJM1307	12.3665	GCA_000977055.2	-	-	-	Target
52	S. cerevisiae YJM1311	12.5492	GCA_000977085.1	-	-	-	Target
53	S. cerevisiae YJM1326	12.4065	GCA_000977115.2	-	-	-	Target
54	S. cerevisiae YJM1332	12.2673	GCA_000977145.1	-	-	-	Target
55	S. cerevisiae YJM1336	12.2673	GCA_000977175.2	-	-	-	Target
56	S. cerevisiae YJM1338	12.2186	GCA_000977205.1	-	-	-	Target
57	S. cerevisiae YJM1341	12.5251	GCA_000977235.1	-	-	-	Target
58	S. cerevisiae YJM1342	12.6226	GCA_000977265.2	-	-	-	Target
59	S. cerevisiae YJM1355	12.943	GCA_000977295.1	-	-	-	Target
60	S. cerevisiae YJM1356	12.5725	GCA_000977325.1	-	-	-	Target
61	S. cerevisiae YJM1381	12.4062	GCA_000977355.2	-	-	-	Target
62	S. cerevisiae YJM1383	12.5861	GCA_000977385.1	-	-	-	Target
63	S. cerevisiae YJM1385	12.9205	GCA_000977415.1	-	-	-	Target
64	S. cerevisiae YJM1386	12.6155	GCA_000977445.1	-	-	-	Target
65	S. cerevisiae YJM1387	12.821	GCA_000977475.2	-	-	-	Target
66	S. cerevisiae YJM1388	12.5221	GCA_000977505.1	-	-	-	Target
67	S. cerevisiae YJM1389	12.2528	GCA_000977535.1	-	-	-	Target
68	S. cerevisiae YJM1399	12.5094	GCA_000977565.1	-	-	-	Target
69	S. cerevisiae YJM1400	12.3575	GCA_000977595.1	-	-	-	Target
70	S. cerevisiae YJM1401	12.1686	GCA_000977625.1	-	-	-	Target
71	S. cerevisiae YJM1402	12.5655	GCA_000977655.1	-	-	-	Target
72	S. cerevisiae YJM1415	12.9341	GCA_000977685.1	-	-	-	Target
73	S. cerevisiae YJM1417	12.703	GCA_000977715.3	-	-	-	Target
74	S. cerevisiae YJM1418	12.4463	GCA_000977745.1	-	-	-	Target
75	S. cerevisiae YJM1419	13.5384	GCA_000977775.2	-	-	-	Target
76	S. cerevisiae YJM1433	12.5256	GCA_000977805.1	-	-	-	Target

77	S. cerevisiae YJM1434	12.5479	GCA_000977835.1	-	-	-	Target
78	S. cerevisiae YJM1439	12.9281	GCA_000977865.1	-	-	-	Target
79	S. cerevisiae YJM1443	12.0458	GCA_000977895.1	-	-	-	Target
80	S. cerevisiae YJM1444	12.2635	GCA_000977925.1	-	-	-	Target
81	S. cerevisiae YJM1447	12.2163	GCA_000977955.1	-	-	-	Target
82	S. cerevisiae YJM1450	12.8449	GCA_000977985.1	-	-	-	Target
83	S. cerevisiae YJM1460	12.6569	GCA_000978015.1	-	-	-	Target
84	S. cerevisiae YJM1463	13.3898	GCA_000978045.1	-	-	-	Target
85	S. cerevisiae YJM1477	12.1849	GCA_000978075.2	-	-	-	Target
86	S. cerevisiae YJM1478	12.7035	GCA_000978105.1	-	-	-	Target
87	S. cerevisiae YJM1479	12.4266	GCA_000978135.1	-	-	-	Target
88	S. cerevisiae YJM1526	12.5274	GCA_000978165.1	-	-	-	Target
89	S. cerevisiae YJM1527	12.7146	GCA_000978195.1	-	-	-	Target
90	S. cerevisiae YJM1549	13.2166	GCA_000978225.1	-	-	-	Target
91	S. cerevisiae YJM1573	12.2409	GCA_000978255.1	-	-	-	Target
92	S. cerevisiae YJM1574	12.2871	GCA_000978285.1	-	-	-	Target
93	S. cerevisiae YJM1592	12.3227	GCA_000978315.1	-	-	-	Target
94	S. cerevisiae YJM1615	12.7047	GCA_000978345.1	-	-	-	Target

2. Pantools homology group validation

To see whether or not the annotation transfer algorithm transfers genes to the correct location in the target genome, validation was done using Pantools' build in homology grouping functionality. For this validation, it would be optimal if every gene from the reference genome was in a homology group with either no other genes, indicating that this gene is not present in the other genome, or with a single gene in the other genome, so it is clear where the annotation transfer should happen to. Of course, there might be variation between two homologous genes, which would cause them to be in a different homology group when grouping settings are too stringent. On the other hand, if grouping settings are too relaxed, genes might end up in larger gene families instead of only with their one homologous gene.

To find the optimal parameters in this trade-off, Pantools' 8 precooked group settings (d1 through d8) were tested on 5 pangenomes containing 2 genomes (*E. coli* strains, *S. cerevisiae* strains, *Drosophila* species, *A. thaliana* accessions and *A. thaliana* assembly versions) and on 2 pangenomes containing 4 genomes of decreasing relatedness (strains/accessions, genera and species of both *Saccharomycetaceae* and *Brassiciaceae*) (See supplementary chapter 4).

For each setting, the number of members in each homology group can be counted and added up, but this would not be informative, as one gene can have multiple mRNA's in one homology group. Therefore, the number of homology groups with exactly 2 contributing genes from both genomes was extracted with a Cypher query shown below and plotted as validatable genes (Fig. S1, S2, S3, S4, S5). For the two pangenomes containing 4 genomes, this query was ran three times with genome numbers 2, 3 and 4 as target genomes instead of only 2 (Fig. S8, S9, S10, S11, S12, S13).

```
match (h:homology_group)-[r:has_homolog]->(m:mRNA)<-[c:codes_for]-(g:coding_gene)
with h, collect(g) as genes
where single(gen in genes where gen.genome=1) and single(gen in genes where gen.genome=2)
with h
match (g1:coding_gene)-[c1:codes_for]->(m1:mRNA)<-[r1:has_homolog]-(h)-[r2:has_homolog]->(m2:mRNA)<-[c2:codes_for]-(g2:coding_gene)
where g1.genome=1 and g2.genome=2
return g1.id as id1, g1 as reference, g2.id as id2, g2 as shouldbe
```

As can be seen from these results, d1 settings (the most stringent settings) resulted in the highest amount of validatable genes in each of the 5 pangenomes. This means that as the group settings become less stringent, more genes shift into homology groups with more than two members than genes that shift out of a homology group with 1 member.

From these data, it was concluded that the most stringent group settings (d1) were best to validate the annotation transfer algorithm with. From this analysis, all genes in a two-member homology group with one gene from every genome will be used to test the precision and recall of the annotation transfer on single copy homologs. Of genes that are in a homology group by themselves, it will be assumed that they should not be transferred to the target genome. Genes that are in a homology group with more than 2 members will be tested to estimate annotation transfer precision and recall with multiple-copy homologs.

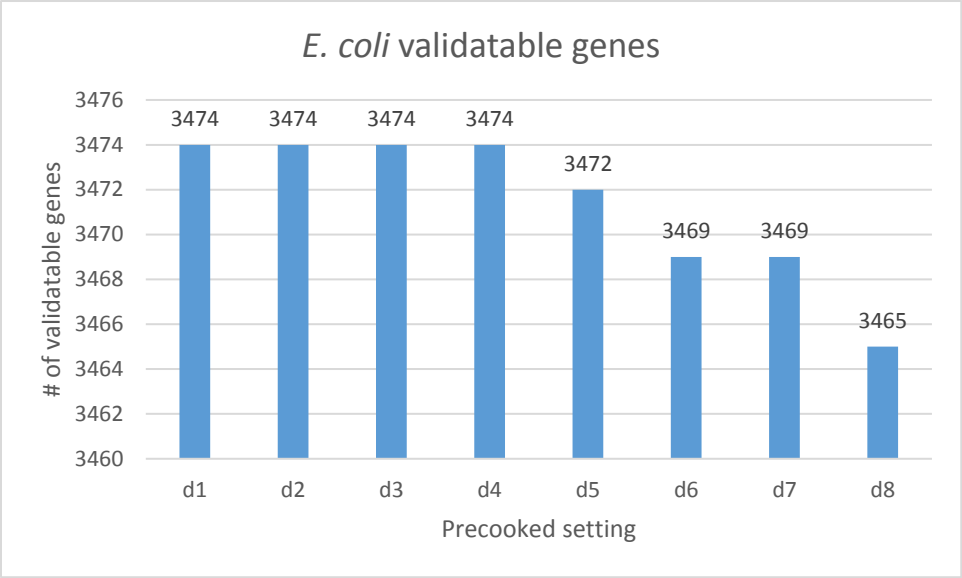

Figure S1 - E. coli *strains validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

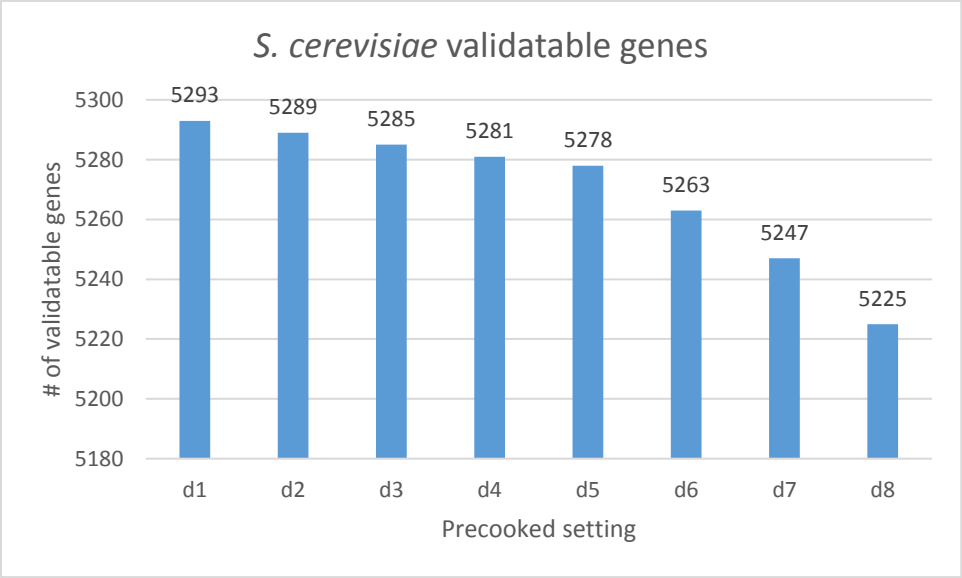

Figure S2 – S. cerevisiae *strains validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

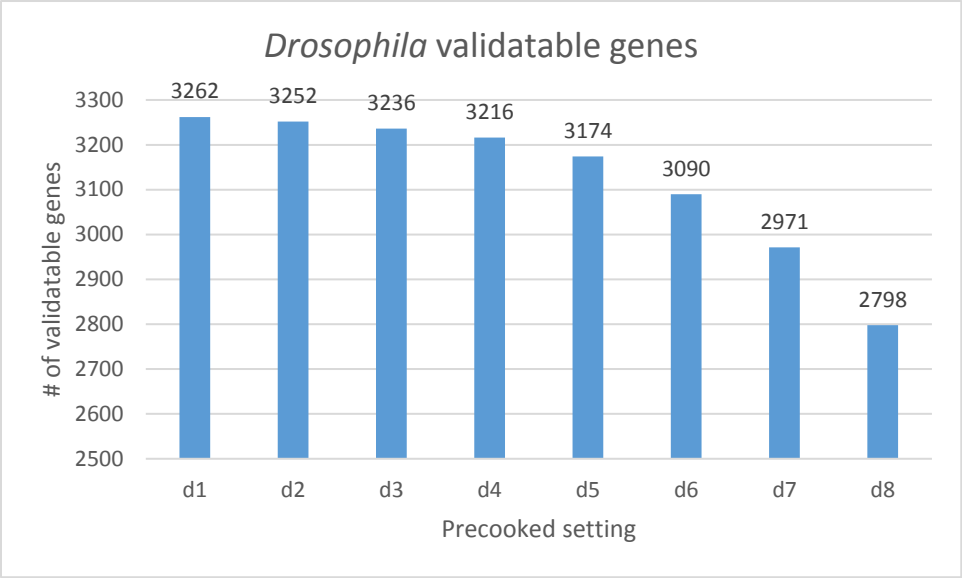

Figure S3 - Drosophila *species validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

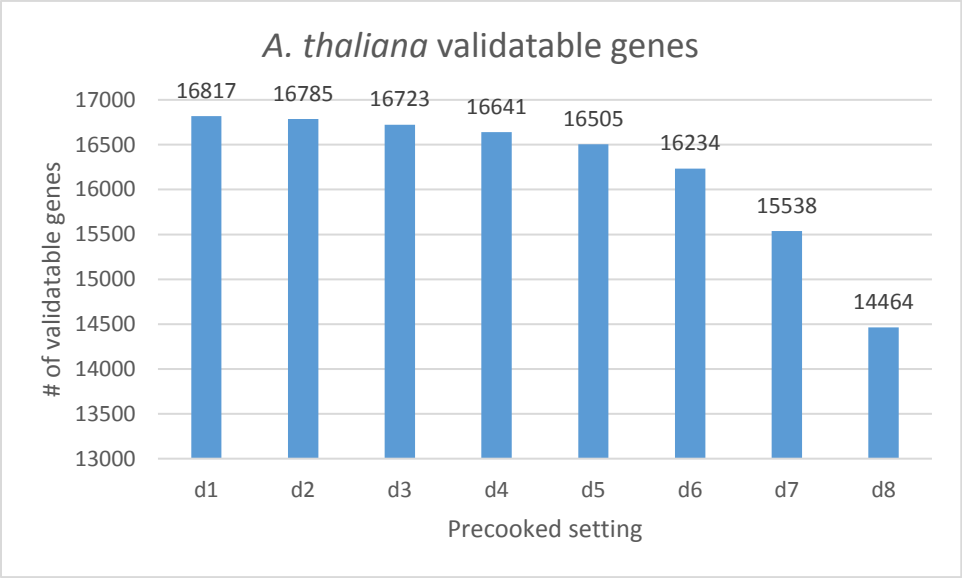

Figure S4 – A. thaliana *accessions validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

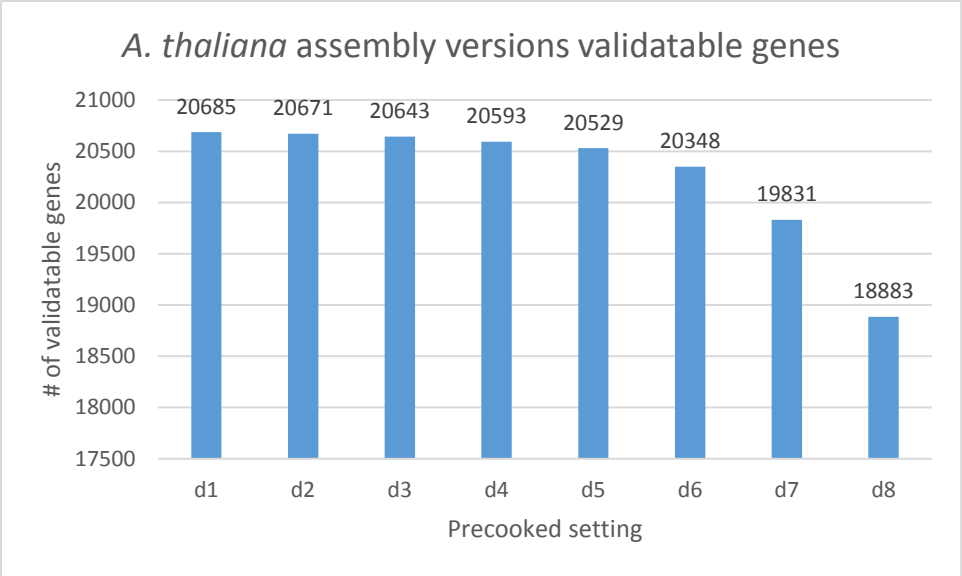

Figure S5 – A. thaliana *assembly versions validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

3. Multiple-copy homolog transfer

To find suitable genes to test the transferring of genes in a gene family with many related genes, all homology groups containing ARF genes in *A. thaliana* were looked at. In all homology groups except for the circled five, it can be seen that there are two genes present per homology group. Two homology groups may be present for two genes when two genes produce more than one mRNA, and those mRNA's are put into separate homology groups. Nonetheless, for all of these genes, it is clear what their transfer target should be.

However, this is not the case for the five circled homology groups. Genes in those homology groups are all similar to each other (as indicated by the is_similar_to edges), and some homology groups even contain more than two genes. Luckily, there are two homology groups that do contain only two genes, despite the fact that those genes are also similar to other genes outside of the homology group. Thus, these genes are good to test annotation transfer specificity with, as these genes can be validated but do provide the risk that they may end up in a wrong position.

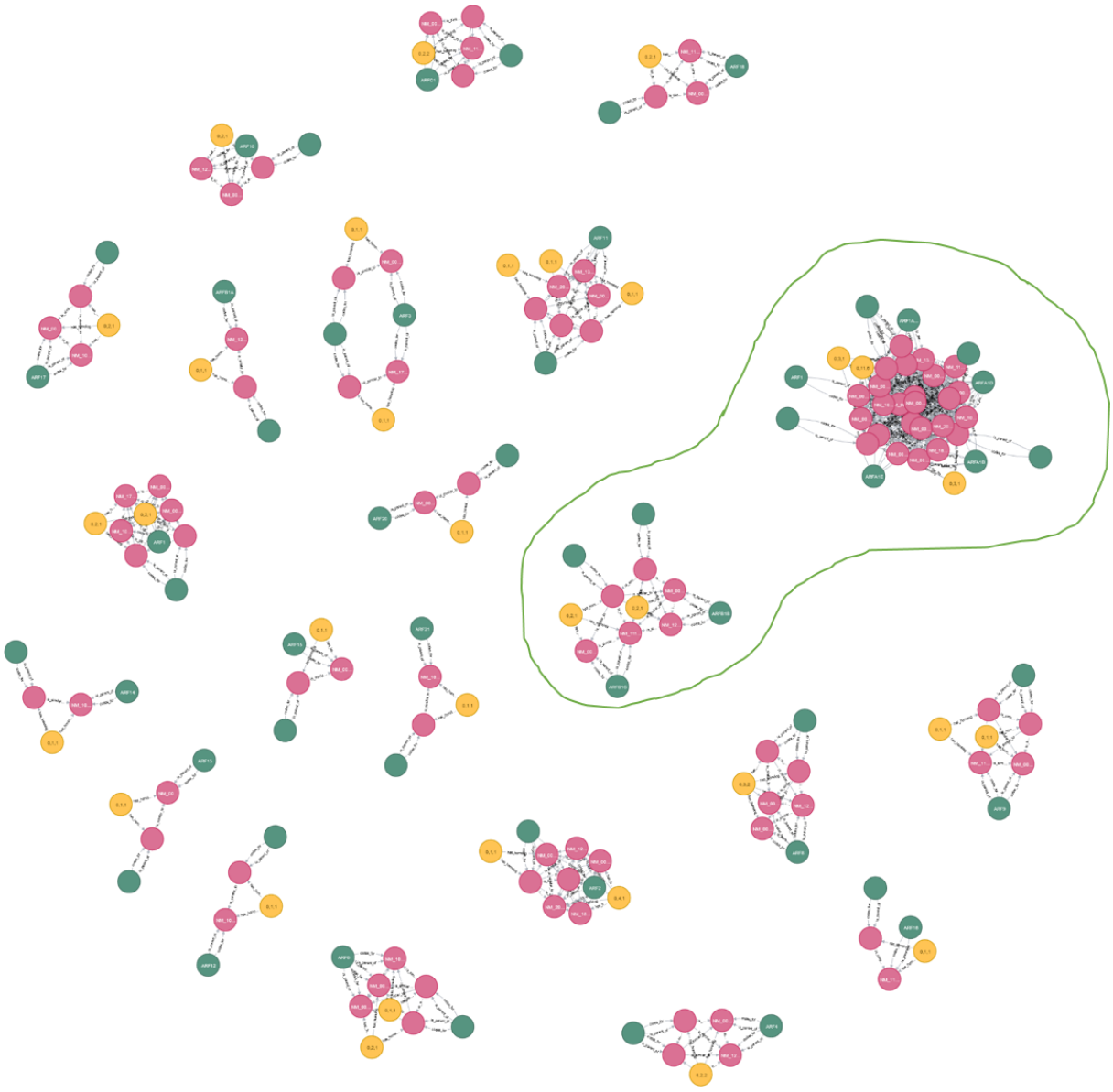

Figure S6 – All homology groups containing ARF's in A. thaliana*. The yellow nodes represent homology groups, the pink nodes represent mRNA's and the green nodes represent genes.*

4. Minimum genome relatedness

It was tried to estimate how related genomes should be for annotation transfer to be a viable strategy to produce a high quality annotation. However, due to time restrictions, this analysis was not finished.

Two pangenomes were made with one reference genome and 3 target genomes each. The three target genomes had a decreasing amount of relatedness to the reference, namely strain/accession, genus and species as can be seen in Table 4.

Table 3 - Minimum genome similarity testing pangenomes

Number	Organism	Genome size (Mb)	Genbank accession	sacc_4
1	Saccharomyces cerevisiae S228C	12.1571	GCA_000146045.2	Reference
2	Saccharomyces cerevisiae YJM993	12.4989	GCA_000662435.1	Target
3	Saccharomyces eubayanus FM1318	11.7342	GCA_001298625.1	Target
4	Naumovozyma castellii CBS 4309	11.2195	GCA_000237345.1	Target

Number	Organism	Genome size (Mb)	Genbank accession	bras_4
1	Arabidopsis thaliana Col-0	119.669	GCA_000001735.2	Reference
2	Arabidopsis thaliana Ler-0	118.891	GCA_000835945.1	Target
3	Arabidopsis lyrata subsp. Lyrata	206.823	GCA_000004255.1	Target
4	Brassica rapa FPsc (B3)	314.865	GCA_003434825.1	Target

Homology grouping was done using all 8 grouping settings as mentioned before. The results of this can be seen in figures S8, S9, S10, S11, S12 and S13.

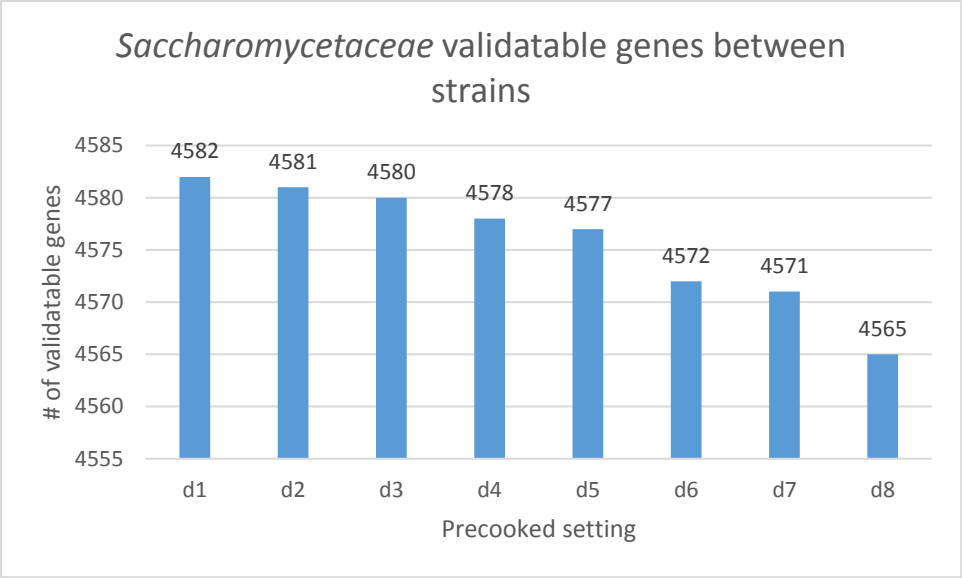

Figure S8 – Saccharomycetaceae *strains validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

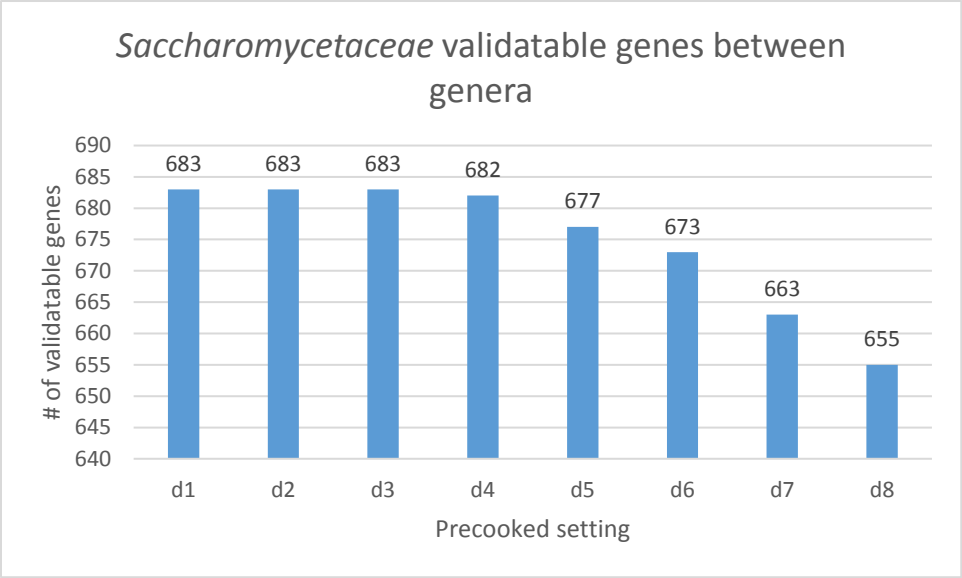

Figure S9 – Saccharomycetaceae *genera validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

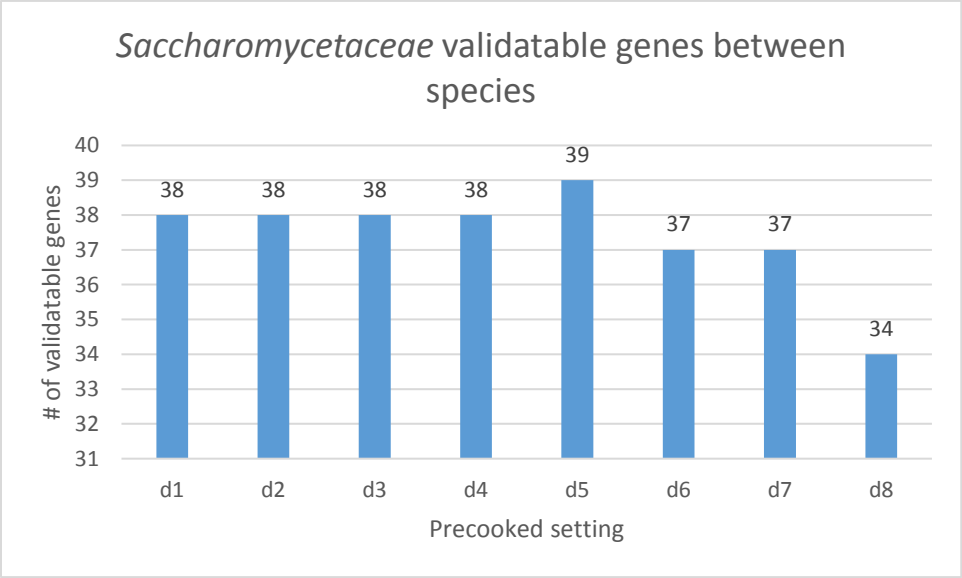

Figure S10 – Saccharomycetaceae *species validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

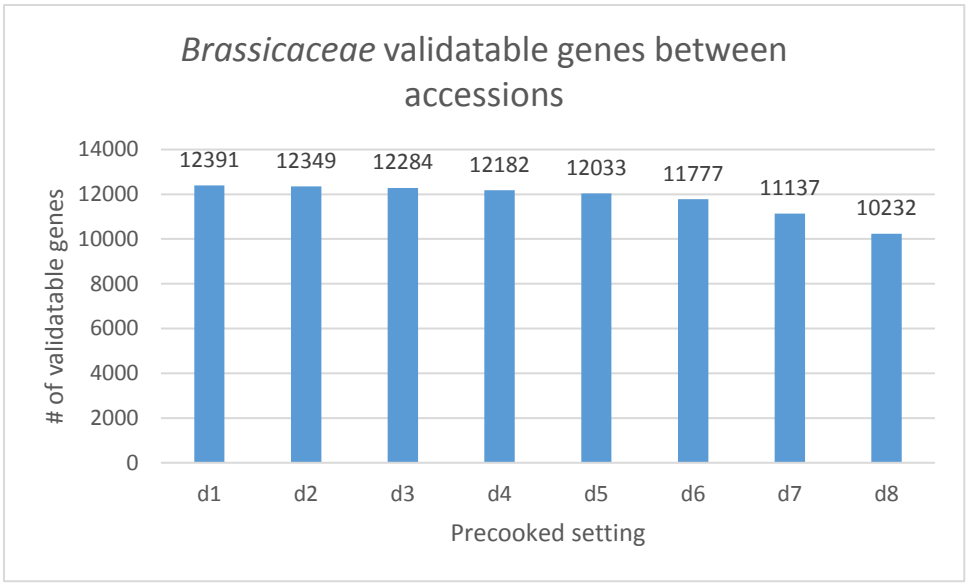

Figure S11 Brassicaceae *accessions validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

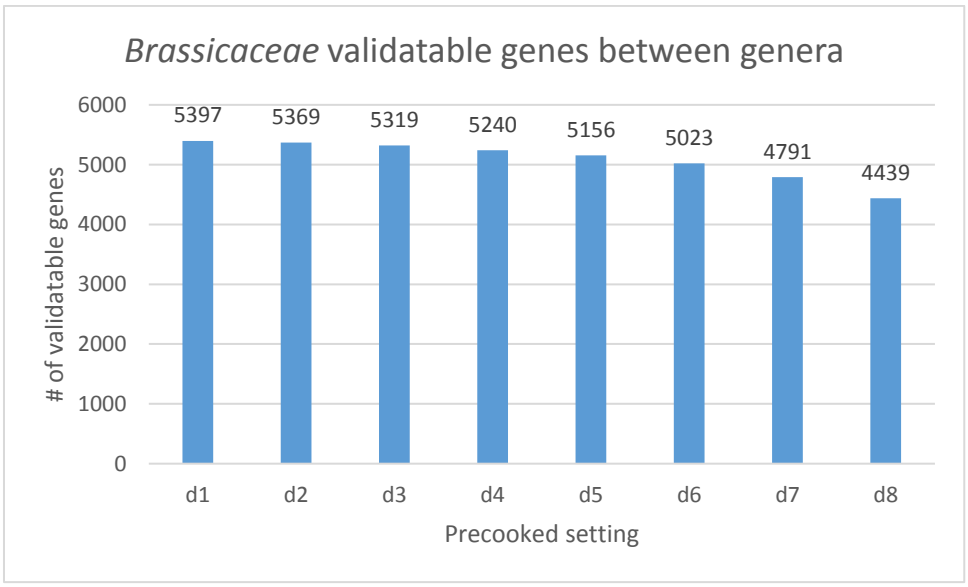

Figure S12 Brassicaceae *accessions validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

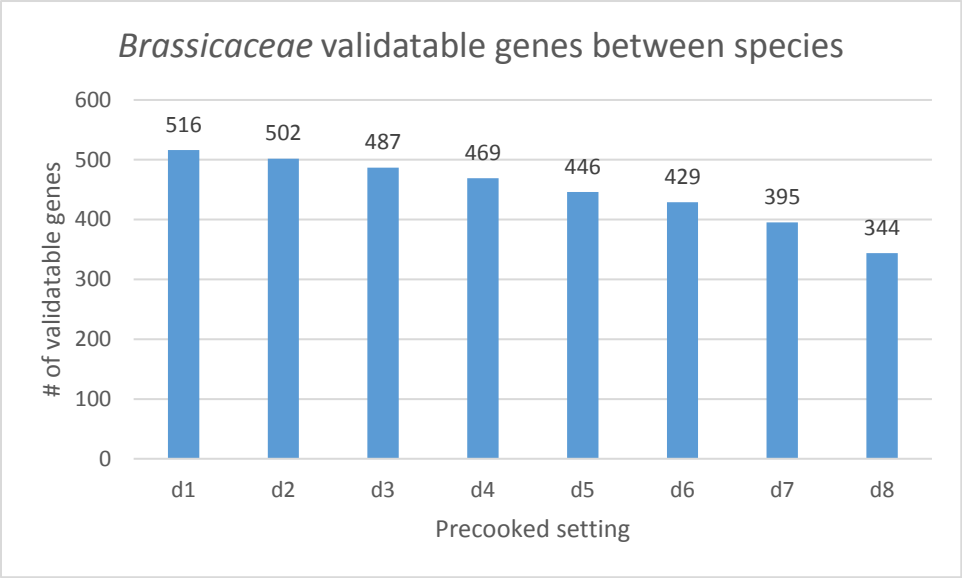

Figure S13 Brassicaceae *accessions validatable genes with Pantools' group settings d1 through d8. On the x-axis, the group setting is shown. On the y-axis, the number of validatable genes (i.e. homology groups with 2 members, one from each genome) is plotted.*

From these results, it was concluded that d1 settings were also best to use in this analysis. However, the number of validatable genes became much lower than expected, and the analysis was aborted at this point. The number of validatable genes can be seen in Figure S14.

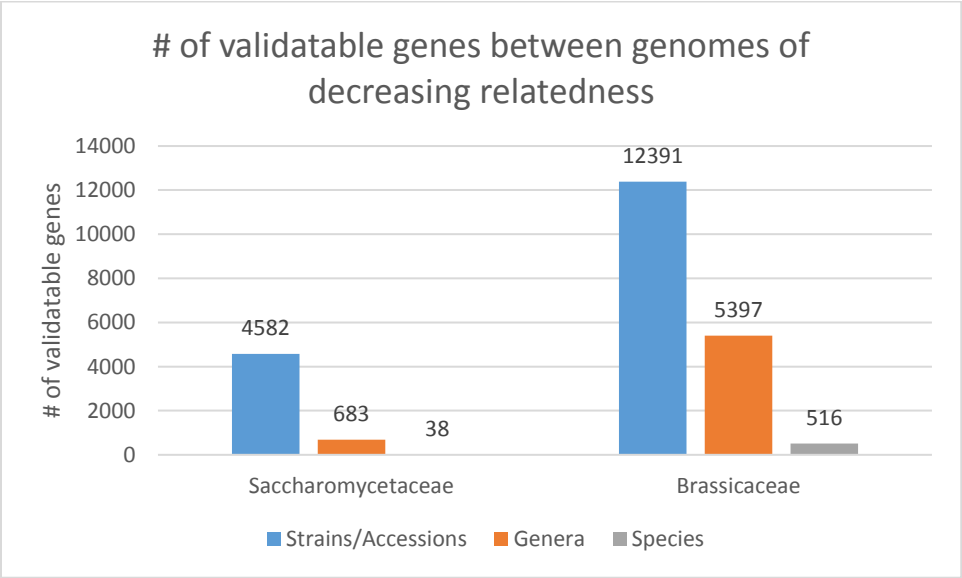

Figure S14 – Validatable genes

5. Single-copy homologous genes transfers

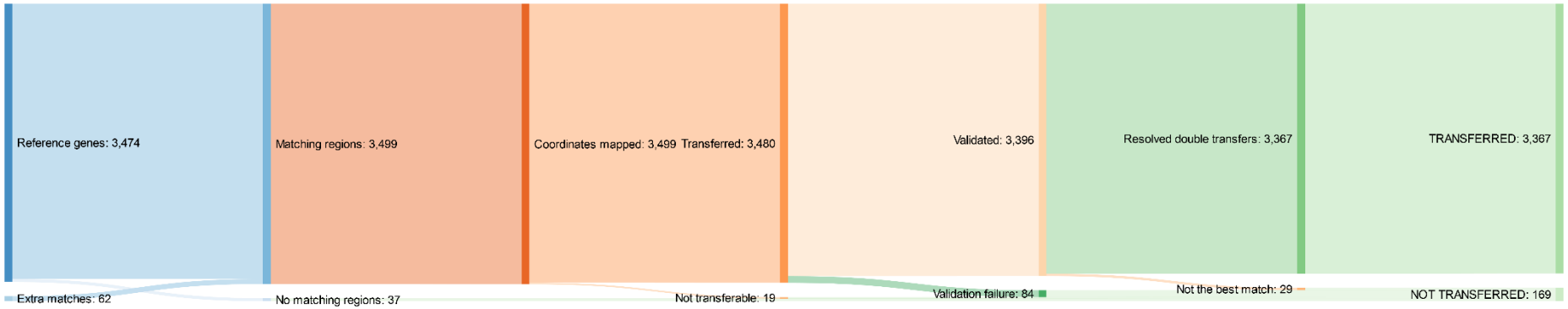

Figure S14 – Single-copy homolog transfer results for E. coli *strains. In the region finding step, for 3437 out of 3474 genes, matching regions are found (98.9%). The other 37 genes had no matching regions. There are also 62 additional matches from genes that had more than one matching region. For all 3499 matching regions, nucleotide coordinates were mapped. In the transfer step, 3480 of the 3499 matches (99.5%) can be transferred successfully. This means that the other 19 matching regions match only part of the gene region and missed either the begin or end location in the coordinate mapping index. When validating the 3480 transferred genes, 3396 (97.6%) pass. For the resolving of double transfers in one region, 3367 out of 3396 (99.1%) validated transfers are the best match in that location. Overall, of all 3474 genes that the algorithm started with, 123 were not transferred (meaning that most dropouts were extra matches), and 16 genes were transferred twice.*

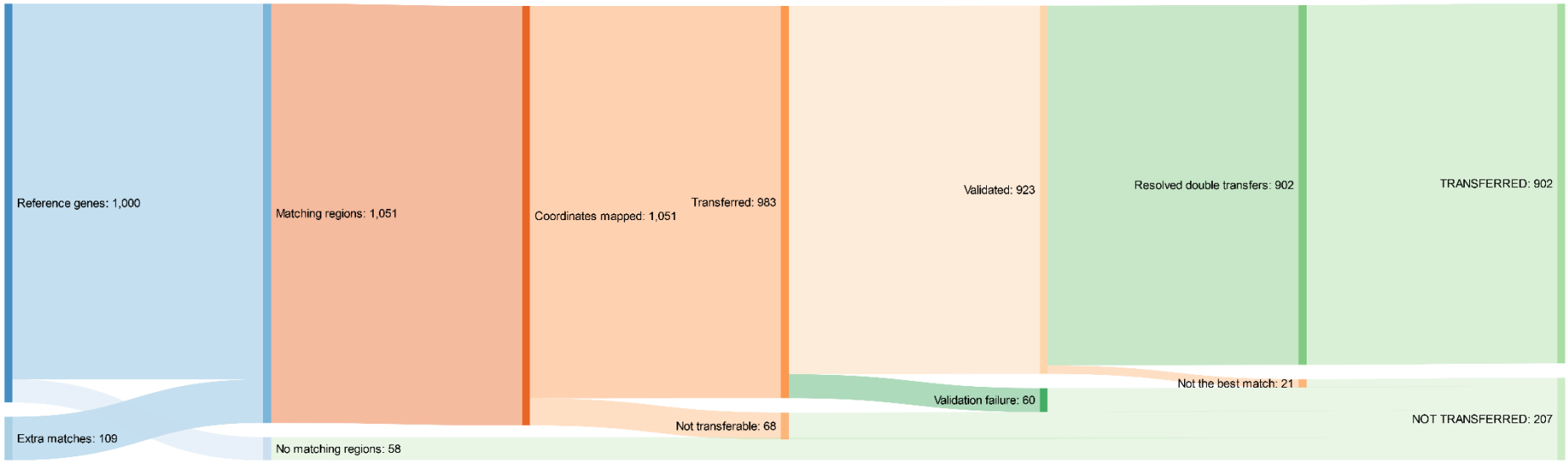

Figure S15 - Single-copy homolog transfer results for A. thaliana *accessions. In the region finding step, for 942 out of 1000 genes, matching regions are found (94.2%). The other 58 genes had no matching regions. There are also 109 additional matches from genes that had more than one matching region. For all 1051 matching regions, nucleotide coordinates were mapped. In the transfer step, 983 of the 1051 matches (93.5%) can be transferred successfully. This means that the other 68 matching regions match only part of the gene region and missed either the begin or end location in the coordinate mapping index. When validating the 983 transferred genes, 923 (93.9%) pass. For the resolving of double transfers in one region, 902 out of 923 (97.7%) validated transfers are the best match in that location. Overall, of all 1000 genes that the algorithm started with, 99 were not transferred (meaning that most dropouts were extra matches), and 1 gene was transferred twice.*

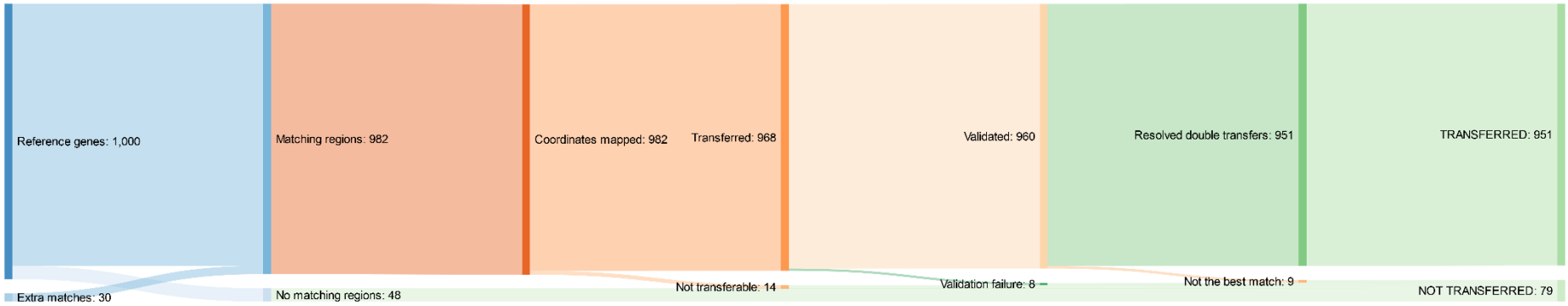

Figure S16 - Single-copy homolog transfer results for A. thaliana *assembly versions. In the region finding step, for 952 out of 1000 genes, matching regions are found (95.2%). The other 48 genes had no matching regions. There are also 30 additional matches from genes that had more than one matching region. For all 982 matching regions, nucleotide coordinates were mapped. In the transfer step, 968 of the 982 matches (98.6%) can be transferred successfully. This means that the other 14 matching regions match only part of the gene region and missed either the begin or end location in the coordinate mapping index. When validating the 968 transferred genes, 960 (99.2%) pass. For the resolving of double transfers in one region, 951 out of 960 (99.1%) validated transfers are the best match in that location. Overall, of all 1000 genes that the algorithm started with, 51 were not transferred (meaning that most dropouts were extra matches), and 2 genes were transferred twice.*

6. Benchmarking

To see how Pantools' annotation transfer compares to existing annotation tools, benchmarking should have been done. Annotation tools that are currently most widely used are *de novo* tools MAKER2 and BREAKER1 and annotation transfer tools RATT and CAT. As RATT and CAT both report better performance than *de novo* tools, the plan was to only benchmark with RATT and CAT. However, ultimately because of limited time, benchmarking couldn't be done.

Even though this is a clear weakness in this research, it is worth mentioning that a lot of fruitless time and effort was in fact spent on getting RATT and CAT to run. However, both tools came with serious problems which in the end, there was no time to deal with.

RATT was easy to install, but it kept on producing an internal error while reading a temporary embl file it wrote itself (given error message, replacing the x and y with genome coordinates: "Probel mwith the embl file at position: x..y)"). As this problem was not related to input files or settings, and there was no documentation on how to fix this error, RATT could not be tested further.

On the other hand, CAT never got to actually be installed and run because of a lot of third party software dependencies, software version requirement conflicts and server type requirement problems. Even though a docker is provided to run CAT inside of, in the end there was no time left to do this.

With this regrettable course of events in mind, it is safe to say that Pantools' annotation transfer wins at ease of use and installation, as it comes built into Pantools itself. The only third party dependency it currently has is Mafft, and this is already planned to be resolved as well, as the newest version of Pantools also has alignment functionality built in.

All in all, benchmarking remains to be done in the future.