Anchors for Homology-Based Scaffolding

Karl K. Käther[1*], Thomas Gatter[1], Steffen Lemke[2], Peter F. Stadler[1,3,4,5,6]

[1*]Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, Leipzig University, Härtelstrasse 16-18, D-04017 Leipzig, Germany.
[2]Zoology Department, University of Hohenheim, Garbenstr. 30, D-70593, Stuttgart, Germany.
[3]Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany.
[4]Department of Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Wien, Austria.
[5]Facultad de Ciencias, Universidad National de Colombia, Bogotá, Colombia.
[6]Center for non-coding RNA in Technology and Health, University of Copenhagen, Ridebanevej 9, DK-1870 Frederiksberg, Denmark.
[7]Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501, USA.

*Corresponding author(s). E-mail(s): karl@bioinf.uni-leipzig.de;

Abstract

Background: Homology-based scaffolding is the task of ordering contigs based on their co-linearity with the DNA sequence of other, closely related reference genomes. We investigate here how a recent approach to identify synteny anchors can be adapted to this task. The advantage of using anchors is that "sufficiently unique" anchor candidates can be precomputed independently for each genome assembly, including contig-level drafts. This allows an efficient identification of co-linear runs of anchor matches between a contig and a reference genome and, consequently, also order their contigs in relation to the reference. Such data can be improved by aggregating scaffolding results for multiple reference genomes.
Results: We present a prototypical pipeline based on the anchor-based synteny detection tool `AncST` and show that it produces scaffolds with accuracies comparable to competing tools at acceptable computational costs. Assuming that anchor candidates have been precomputed for the genomes of interest, the remaining

computational efforts are on par with the fastest methods. Finally, the results suggest that our approach is particularly well suited for homology-based scaffolding with multiple references.
Conclusions: Homology-based scaffolds can effectively be computed with a greedy-like approach using synteny anchors. The method performs comparably well with advantages regarding scaffold contiguity and multi-reference scaffolding.

Keywords: Genome Assembly, Synteny, Maximum Matching

1 Background

Genome assembly is one of the basic tasks in computational biology. While cost-effective high-throughput sequencing methods readily produce high-quality contigs, arranging them in the correct order and orientation is a much more difficult problem. Assembly from short-read sequencing data typically stops at the level of contigs, i.e., segments of genomic sequence that are covered by sequencing reads with sufficient overlaps to ensure that they represent a single genomic interval. Scaffolding most generally refers to the task of ordering and orienting the contigs in their correct chromosomal order. In many cases, additional data is required for this task. Examples include the sequencing of mate-pairs, i.e., DNA fragments of approximately known length for which only the sequence at their ends are known, optical mapping [1], or Hi-C data [2], which have become the become the de facto gold standard for novel sequencing project [3]. An alternative is homology-based scaffolding in which a closely related reference genome is used as a "template". Since high-quality reference genome assemblies are becoming available at an increasing pace, homology-based scaffolding is becoming an attractive proposition. Several approaches for this task have been proposed in the last few years [4, 5, 6].

Homology-based scaffolding starts by building alignments of the reference and target genomes and then processing the order and orientation information provided by the individual alignments to derive sequences of contiguous scaffolds. For instance, [5] use the `minimap2` sequence aligner [7] and leverage the longest alignments to order the contigs. In `Ragout2` [8] multiple sequence alignments and different levels of synteny blocks are used to create a breakpoint graph. Rearrangement operations are used in [4] to minimize the cost required to turn one genome into the other, thereby defining a minimum cost contig order. Even though most of the available tools use sequence alignments explicitly, it has been shown that so-called ordered minimizer sketches, i.e., selections of k-mers, can also be successfully applied for assembly scaffolding in `ntJoin` [6]. Several tools have been developed to use multiple reference genomes [9, 10, 8]. The scaffolding process may become more robust to species-specific rearrangements and thus reduces biases from using only one reference. Such approaches can make direct use of multiple sequence alignment blocks [8] or create a consensus from single-reference contig orders as in `RagTag` [9] or `CSAR` [10]. Moreover, homology-based methods have also been applied to improve contig length and quality during assembly [11].

Here, we use our recently developed `AncST` approach [12] to produce high-confidence alignments from synteny anchors, which serve as reliable input for a contig scaffolding pipeline. Since `AncST` is developed to assert 1-1 orthology between respective sequence anchors with high confidence, corresponding alignments should serve as a suitable signal for robust homology-based contig scaffolding. Indeed, a simple greedy-like algorithm already recovers correct orders and orientations comparable and partially superior to existing pipelines while promising to be relatively scalable. The new homology-based scaffolding pipeline proposed here exhibits comparably high accuracy. It is a timely development given the massive efforts to close phylogenetic gaps in comprehensive genome sequencing projects, such as the Darwin Tree of Life (DToL),[1], the European Reference Genome Atlas (eERGA),[2] or the Earth Biogenome Project (EBP).[3]

2 Theory

Genome assembly from short-read sequencing data typically stops at the level of contigs, i.e., segments of genomic sequence that are covered by sequencing reads with sufficient overlaps to ensure that they represent a single genomic interval. Scaffolding most generally refers to the task of ordering and orienting the target contigs in their correct chromosomal order. Additional data is required for this task. Examples include the sequencing of mate-pairs, i.e., DNA fragments of approximately known length for which only the sequence at their ends are known, optical mapping, or a closely related reference assembly [1].

Informally, homology-based scaffolding is the task of ordering and orienting the contigs of a draft assembly linearly along the chromosomes of a reference assembly, and to identify comparably rare rearrangement events between the two genomes, see Fig. 1. Such rearrangement events can be detected if two parts of a draft contig match non-contiguous locations on the same reference chromosome or distinct reference chromosomes. Homology-based scaffolding thus can be seen as an optimization problem that penalizes non-colinearity of the contiguous alignments of a contig with the reference chromosomes. A scaffolding of set of contigs can be formalized by assigning to each contig i its order $\pi(i)$ and its orientation $o(i) \in \{-1, +1\}$ in the final assembly. In the most general setting, each particular scaffolding is associated with a score $S(\pi, o)$ that depends on order and orientation of the scaffolds. The scaffolding problem then consists in maximizing $S(\pi, o)$. The key idea is to consider the best matching homologous intervals $\ell(i)$ and $\ell(j)$ of contigs i and j in the reference assembly. Positive score contributions are then incurred if $\ell(i)$ and $\ell(j)$ are adjacent on the reference and $\pi(i)$ and $\pi(j)$ are consecutive in the target (with matching orientations). A simplified version of the problem has been shown to be NP-hard [13]. Available tools thus utilize computationally efficient heuristics to assemble contigs into larger scaffolds successfully.

[1] https://www.darwintreeoflife.org
[2] https://www.erga-biodiversity.eu
[3] https://www.earthbiogenome.org

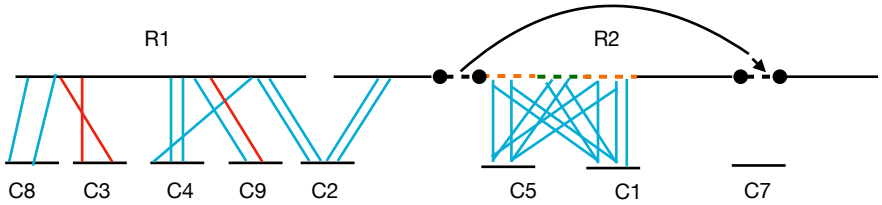

Fig. 1 Homology-based scaffolding. A list of contigs C1 to C8 is ordered along reference chromosomes R1 and R2 based on alignments indicated by colored lines between them. The contigs are initially unordered and placed into the order with respect to their alignments. Here, the `AncST`-based pipeline's strategy is depicted in which a contig gets placed into the order as soon as the majority of its alignments with its best-matching reference chromosome has been seen. Note that not all contigs have to be placed (C6 is missing). Different complications for homology-based scaffolding are shown: Blue lines indicate forward and red ones reverse matches; thus C3 is a contig which should either be turned or marked with one or more inversion events. C2 depicts the case when initial contigs get incorrectly fused in the assembly process which leads to one long scaffold consisting of all contigs aligning to R1 and R2. In R2 the arrow indicates a translocation event and the dashed colored lines denote local duplication events (colors do not matter). Hence, C5 and C1 are in a duplicated region which is difficult to reconcile and C7 is placed incorrectly.

Homology-based scaffolding is closely related to synteny analysis, i.e., the identification of co-linear regions between genomes [14]. The main difference is that the detection of syntenic regions concerns the identification of maximal runs of co-linear alignments, while homology-based scaffolding assumes that one or both genomes are given in the form of short intervals, the contigs, which are assumed to be "almost" always internally co-linear with the other genome. A key problem is therefore to determine whether or not query contigs are adjacent on the reference. Unambiguous 1-1 matches between a query contig and a reference chromosome therefore provide valuable information about the placement of a query contig on the reference. The main contribution of this work is to show that the systematic use of *synteny anchors* (a specific class of anchors introduced in [15, 12] that can be computed efficiently) improves efficiency and accuracy of homology-based scaffolding. Moreover, we show that our approach is particularly suitable for homology-based scaffolding with multiple references.

3 Implementation

Anchors

In recent work [15, 12] we investigated the notion of annotion-free anchors as a means to identify syntenic regions efficiently. As described in full detail in [15, 12], two d_0-unique subsequences w and w' in different genomes in G and G' are guaranteed to form a unique match if their edit distance is less than $d_0/2$. In practice, it suffices to use a cutoff of d_0 minus some small tolerance. We call such d_0-unique sequences anchor candidates. Anchor candidates that satisfy the matching criterion form anchors between the genomes G and G'. Pragmatically, it is preferable to use `blast` bit scores directly. In previous applications we found that considering a pair of anchor candidates x and x' from genomes G and G' as a match yield excellent performance if $s(x, x') > \max(s_0^G(x) + \sqrt{s_0^G(x)}, s_0^{G'}(x') + \sqrt{s_0^{G'}(x')})$, where $s_0^G(x)$ and $s_0^{G'}(x')$ are scores of x and x', resp., against the best non-overlapping matches in the genomes of origin.

Anchors were used in [12] to construct fine-grained synteny maps between not-too-distant genomes. In order to determine anchor candidates efficiently, we use infrequent k-mers with a length close to $\log_4 |G|$, where $|G|$ denotes the genome size, as a indicator for the uniqueness of w. Details of the statistics implemented in `AncST` are described in [12].

Contig sorting

The basic idea underlying our approach is rather straightforward: Since anchors between target contigs and reference chromosomes establish a strict 1-1 orthology relationship, they also anchor contigs at specific locations of the reference assembly. The same 1-1 relationship also justifies a simple greedy approach. Thus, we first determine, for each contig, the reference chromosome with the largest number of matched anchors. This step is designed to remove residual noise introduced by a small fraction of spurious anchor candidates as well as occasional genome rearrangements between target and reference genome. In the current implementation we only attempt to construct homology based scaffolds that match within chromosomes of the reference assembly, i.e., we do not attempt to identify such genome rearrangements between reference and target explicitly. We shall return to this point in the Conclusions section, however. In the second step, we therefore consider each reference chromosome separately. Let $\mathcal{L}$ be an initially empty list. We traverse the anchors between the reference chromosome and its assigned contigs in the order of the reference. For each contig, we keep count of the number of its anchors that have been visited so far. A contig c is appended to $\mathcal{L}$ as soon at least half of its anchors have been visited. The order of their insertion into $\mathcal{L}$ therefore determines the order of the contigs in the emerging scaffold. As we shall see in the benchmarking data below the use of `AncST` anchors, i.e., nearly perfect 1-1 orthologs, reduces noise and ambiguities. It also makes the greedy approach a natural choice since overlapping matches of contigs to the reference are expected to be absent in theory and extremely rare in practice. This simple greedy strategy for determining a contig order relative to a reference assembly can easily be extended to multiple reference assemblies by simply running anchor detection and greedy ordering independently for each reference assembly. Different reference assemblies will yield slightly different contig orders; it then remains to reconcile the conflicting order information.

Merging of alternative scaffolds

Scaffolds obtained with different reference assemblies can be performed using a simple graph-theoretic approach [10]. To this end, we define a graph $\Gamma = (V, E, w)$ whose vertex set are the end points of the target contigs, i.e., each target contig is represented by *two* vertices. An edge $\{u, v\}$ is inserted if u and v are endpoints of two distinct scaffold that are consecutive for one of the reference assemblies. Note that for a pair of consecutive contigs the choice of the adjacent endpoints determines the relative orientation of the two contigs. Moreover, we assign a weight w with the edge $\{u, v\}$ that in the simplest case counts the number of reference assemblies in which u and v are consecutive. More sophisticated weights w can be introduced that account e.g. for the quality of anchors for the contigs of which u and v are endpoints. These weights are simply aggregated if an edge appears for more than one reference assembly. For

technical reasons, it is useful, moreover, to assign edges with weight 0 to all pairs of vertices that are neither consecutive w.r.t. to at least one reference assembly nor form the endpoints of the same contig.

In order to extract a consensus scaffold from the weighted graph Γ, we observe that in a scaffold each contig has at most one predecessor and one successor and thus each end-point of a scaffold can be incident to at most one edge. Hence scaffolds that only contain observed adjacencies correspond to matchings in Γ. A maximum weighted matching in Γ thus serves as a best estimate for a consensus scaffolding. We note that a matching will be perfect, i.e., includes all vertices, only if all contigs are placed into circular chromosomes. Otherwise, there will be contig ends that correspond to ends of a scaffold and thus are not incident to any edge. The maximum weighted matching problem can be solved in polynomial time using the *blossom* algorithm [16].

Adding missing edges to Γ with weight 0 transforms the problem into a maximum weight (or minimum cost) *perfect matching* problem, which can be solved by the highly optimized `Blossom V` implementation [17]. This approach has been pioneered in [10]; we refer to Figure 2 in this publication for a detailed visual description of the algorithm.

In the resulting perfect matching, head and tail of the same contigs are connected, resulting in a regular graph Υ of vertex degree 2, i.e., a vertex disjoint union of cycles. After breaking each cycle by removing its minimum weight edge and removing completely unsupported edges with weight 0, we are left with a disjoint union of paths, each of which specifies (up to the overall orientation) a linear order of contigs as well as the reading direction of each contig, and thus a scaffold.

4 Computational Methods

Synthetic test data

We produced the three simulated draft assemblies of the human genome as follows: For the rearranged version, we first applied rearrangement events (tandem duplications, translocations and inversions). Locations of rearrangements were restricted to respecting annotation items, i.e. breakpoints were only introduced at the start or end of annotated items in the annotation file of the human genome. We distinguish between short-range and long-range rearrangements. The former only affect a single annotation item, while the latter pertain to an interval containing two or more annotation items separated by not more than 1 million bases. In addition to a local inversion of the interval, rearrangements were defined as follows: For translocation we randomly selected a different chromosome, an insertion position, and the orientation of the inserted fragment. Up to four duplication events are allowed, either local or remote. For a local duplication the copy of a selected interval is inserted, possibly in inverted orientation, at a location between 1000 and 100000 bases downstream of the interval in question. Alternatively, duplicates were inserted in a randomly chosen chromosome. If a translocated sequence was chosen to only be translocated but not copied, then the original source interval was deleted in order to avoid bloating the genome size. We created one version in which at least 5% of the human genome are rearranged and another one in which at least 10% are rearranged. The number of duplication events as well

as sequence intervals are chosen arbitrarily. The rationale behind the moderate number of rearrangement events in the test data is that contigs with a large number of rearrangements relative to the reference cannot be safely placed into a scaffold since homology-based scaffolding only can be employed when co-linearity is the norm. The rearranged genomes are documented in the supplementary material.

Contigs were extracted in the same manner from both the rearranged and conserved human genome. In order to mimic the fact that contig assembly usually stops at repetitive sequences, we defined contig ends at stretches of at least 10000 N in the hard-masked version of the human genome. In addition, we introduced random breakpoints within the initial fragments. At this stage we only accepted contigs consisting of at least 20000 bases and 30% unmasked sequence. Each initial fragment was subjected to 30 repetitions of the random fragmentation attempts. In order to simulate assembly errors [18], we fused 5% of the resulting fragments at random. Moreover, reading direction was determined at random.

The genomes assemblies of chimpanzee (Clint_PTRv2/panTro6) and bonobo (Mhudiblu_PPA_v0/panPan) were used as references for scaffolding, while the human genome (GRCh38/hg38) was used as a basis for the synthetic test data sets. All sequence data were downloaded from the UCSC genome browser[4].

Real-life test data

In addition to synthetic data which allows a complete evaluation of the results, we employed our scaffolding pipeline to a set of 20 `Drosophila` published on NCBI (Table 4 in Additional File 3). Here we focus on the application scenario in which genomes of multiple related species are available for homology-based scaffolding. Thus, we arbitrarily chose a genome of lower quality (labelled as `scaffold` or `contig` assembly level on NCBI) for each of the species and scaffolded them with regards to the other 19 species.

Scaffolding

Using our synthetic test data, we compared our approach with some widely used homology-based scaffolding approaches:
`CSAR` and `Multi-CSAR` were installed from their sources[5]. Default parameters were used with the option `--nuc` to run on the nucleotide level on the two pairs of genomes as well as both reference genomes for the multi-reference version.
`RagTag` was installed from its source[6] and run on the two pairs of genomes. To obtain a multi-reference version we used its `merge` command on the result of the pairwise results. Default parameters were used. For the real test data `RagTag` was run with the `minimap` setting *asm 20* to increase sensitivity as we compare genomes of considerable divergence. `ntJoin 1.15` was installed using the respective `bioconda` recipe and was run with the following options to ensure comparability of the results: `agp=True no_cut=True time=True overlap=False`. `Ragout 2.3` was installed using the respective `bioconda` recipe and was run with the following options: `--refine`

[4] https://hgdownload.soe.ucsc.edu/goldenPath/
[5] https://github.com/ablab-nthu/CSAR, https://github.com/ablab-nthu/Multi-CSAR
[6] https://github.com/malonge/RagTag

`--solid_scaffolds`. Multiple sequences alignments were generated with `cactus 2.9.9` and used in `maf` format as input for `Ragout2`.
`AncST` was used to produce pairwise alignments between the respective genomes. The following parameters were used because they initially worked satisfactorily well and no additional time was used to optimize for the data set at hand. The pipeline relies on `GenMap` to produce k-mer statistics. For the experiments with synthetic data (primates), the following parameters were used. The pipeline relies on `GenMap` to produce k-mer statistics and parameters $k = 16$ and $e = 0$ were set. The initial window size for anchor construction was set to 500 bp, with intervals overlapping at their midpoint. Windows with an aggregate k-mer count less than the 7th percentile were used as anchor candidates. The `Drosophila` genomes were processed at the `AncST` webserver at http://anchored.bioinf.uni-leipzig.de:8080/ with default parameters. For both data sets, the resulting pairwise scaffolds were reconciled using a reimplementation of the ideas behind the merging algorithm in `Multi-CSAR` [10] as described in the Computational Methods section. As a weight for each single-reference scaffolding result we use the total `blast bit score` of all `AncST` alignments times 1000 divided by the length of the respective reference genome (to produce weights in a range of few dozens to few hundreds for human-readability). Accordingly, for each contig pair connected in one of the single-reference scaffolds, the produced weight is added to the corresponding edge in the superimposed graph for multi-reference scaffolding.

`RagTag`, `ntJoin` and `Ragout2` do not produce weights. Since weights may have a significant impact on the results, however, we ran the multi-reference experiment with `ntJoin` and `RagTag` of scaffolding the chopped but not further rearranged version of the synthetic human genome with the weights used by our pipeline and noted that the results were exactly the same as for unit weights. Furthermore, for experiments with real data we show in Additional File 6 that using weights from our pipeline in `RagTag` and `ntJoin` does not significantly alter the results. `Ragout2` does not use weights but a multiple sequence alignment produced by `cactus` for which a weighted initial phylogeny can be provided.

Evaluation

In order to compare the different scaffolding results we employ a Needleman-Wunsch global alignment algorithm [19] to quantify the edit distance between reconstructed scaffolds and the correct order of the contigs in the reference. A score of 1 is used for matching contigs, while insertions, deletions, and mismatches are scored with 0. The total score of the optimal alignment therefore equals the number of matched contigs for an optimal solution and, after determining the correct orientations of the aligned contigs, can directly be used as a performance measure. More precisely, we compute the *precision* as the percentage of the Needleman-Wunsch alignment score divided the maximally achievable score, which is is given by the total number of contigs in any rearranged contig minus 1/2 times the number of fused contigs. Alternatively, we may normalize the Needleman-Wunsch alignment score by the total number of contigs that are placed into any scaffold, resulting in a measure that is akin to *specificity*. Throughout, we refer to these metrics as *NW rec.* and *NW prec.*, respectively.

Moreover, we also compute the total length of DNA covered by the correctly aligned (and oriented contigs, i.e. exactly those with positive contribution in the NW alignment, normalized by the total length of all contigs from the original genomes (abbreviated as *NW DNA cov.*).

In addition to evaluating the alignments, we count the number of correctly connected contigs among all possible correct connections (*precision*) and all made connections (*specificity*). In other words, the number of correctly predicted ***adj**acent* contig pairs is related to the number of respectively true pairs (*adj. prec.*) or all reported scaffold pairs (*adj. spec.*). Similar as for NW, we also compute the total length of DNA covered by correctly connected contigs relative to genomic length. Here, exactly all contigs are included that are connected correctly on at least one side, even if they are also involved in a false connection. We further refer to this metric as *adj. DNA cov.* Finally, we record the number of *breaks* which are introduced to the original chromosomes when they are covered by more than one newly assembled scaffold.

Moreover, we employed the widely used assembly evaluation tool `Quast` (`v5.3.0`) [20] and compare the contiguity statistic *auN* (`a`rea `u`nder `N`x curve, see [21]) for which a higher value indicates better performance. Additionally, we consider the number of *misassemblies* (for which a low number is preferable) reported against the respective reference genomes for both the synthetic and real data sets. We employ `auNA` as a measure of contiguity which takes into consideration only correct alignment blocks with the reference genome.

5 Results

Synthetic Data - Custom Metrics

We employed five different scaffolding pipelines for comparison: `CSAR` [4] and their respective multi-reference version `Multi-CSAR` [10], `RagTag` [9] with their multi-reference `merge` command, `ntJoin` which allows multi-reference scaffolding as a default, `Ragout2` with `cactus` multiple alignments, and our pipeline based on `AncST` anchors [12]. We used `Ragout2` only for the multi-reference version of the experiments since the requirement of generating the respective pairwise alignments with `cactus` was deemed too resource-intensive and the tool was developed specifically for the use of multiple sequence alignments for homology-based scaffolding with mutiple references [8].

Table 1 summarizes the performance results for scaffolding the contigs of the human genome produced by breaking it up at regions of considerable repeat content and length as well as reversing and fusing some of the resulting contigs.

Tables 2 and 3 show the results for the rearranged assemblies. The details of the construction of the synthetic target assemblies can be found in the respective Computational Methods section.

We decided to note the time consumption as wall-clock time for the different runs we present here for simplicity. We note that `CSAR` and `RagTag` were run as single-core applications while we used 58 cores for `AncST` anchor computation. `CSAR` cannot be run multi-threaded although the consumed `user` time indicates use of multiple cores. We decided not to benchmark `RagTag`, `Ragout2` and `ntJoin` with multiple threads as

they are so fast that time consumption is negligible already without multithreading. Multiple alignment computation with `cactus`, however, was done using 64 available cores.

Table 1 Comparison of scaffolding performance. Data refer to the synthetic contig sets obtained by fragmenting the human genome. The genomes of chimpanzee (tro) and bonobo (pan) are used as references. We refer to Section 4 for details on these metrics. The second columns for the alignment-based statistics in brackets considers the case in which only one new scaffold is allowed to be aligned to each reference chromosome as opposed to the figure before the brackets for which multiple scaffolds can align to each reference chromosome. All values (except for time measurements and the number of chromosome breaks) are percentages. Some of the best and worst performances are highlighted for each column (values in brackets are considered as separate columns except for the second time one). Accordingly, red means worst performing, magenta is within 3 percentage/time/count points of the worst, blue means best performing and cyan is within 3 percentage/time/count points of the best. Execution times are approximate wallclock times on a Linux server with 256 GB RAM and 64 available cores on two Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz processors. The second time column reports the running time of (1) `AncST` without the pre-computation of anchors for each genome, since the latter can be performed as a one-time preprocessing for each genome and (2) `cactus` for `Ragout2`.

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3200	65 (33)	63	64 (35)	94	90	91	83
CSAR tro	3200	63 (31)	62	59 (28)	95	91	91	75
Multi-CSAR	6500	87 (3)	82	86 (5)	43	68	70	1086
RagTag pan	25	79 (77)	82	80 (76)	93	90	87	9
RagTag tro	25	82 (80)	84	83 (79)	92	89	86	5
Multi-RagTag	52	60 (9)	92	80 (11)	48	86	75	327
ntJoin pan	20	79 (77)	81	80 (74)	92	88	82	10
ntJoin tro	20	82 (80)	83	82 (79)	92	87	82	4
Multi-ntJoin	28	89 (28)	96	77 (19)	85	93	74	178
Ragout2 (multi)	4100 (15)	74 (39)	80	79 (38)	93	97	91	58
AncST pan	750 (14)	77 (75)	85	80 (76)	89	92	87	6
AncST tro	750 (14)	80 (78)	88	82 (78)	90	92	88	4
Multi-AncST	1500 (30)	78 (76)	84	79 (75)	90	92	88	2

58 cores were used for `AncST` anchor computation. 64 cores were used for `cactus` computation. All other applications were run with default parameters, i.e., one core for `RagTag`,`ntJoin` and `Ragout2`. Whether `CSAR` uses multiple cores does not seem to be documented.

In general, all tools display certain strengths and weaknesses. `CSAR` takes the longest by far and is able to make mostly right contig connections. On the other hand, its NW-alignments results show that in the single-reference scenario larger scaffolds are created with a significant amount of wrong connections. In a multi-reference scenario it creates many small scaffolds and performs disappointingly. `RagTAG` and `ntJoin` both perform relatively well overall and particularly in single-reference scenarios. In multi-reference scenarios they both produce rather small scaffolds as well. `Ragout2` shows decent overall performance as well and makes most correct contig connections. Finally, our pipeline finds itself in the midfield of most of these performance measures. Its strengths are clearly the production of relatively large scaffolds with mostly correct contig connections and robustness for multi-reference scenarios.

Table 2 Scaffolding results for the chopped human genome with at least 5% rearranged sequence. For details we refer to the caption of Table 1

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3200	50 (17)	53	50 (19)	82	85	83	156
CSAR tro	3200	53 (19)	56	56 (20)	83	83	85	161
Multi-CSAR	6500	79 (3)	80	75 (3)	37	63	60	1692
RagTag pan	28	67 (64)	79	73 (69)	78	86	79	24
RagTag tro	26	71 (67)	83	76 (72)	78	86	79	18
Multi-RagTag	56	43 (5)	89	71 (7)	31	78	65	460
ntJoin pan	20	68 (65)	81	72 (69)	79	89	79	17
ntJoin tro	20	72 (68)	85	76 (72)	79	88	78	13
Multi-ntJoin	28	77 (20)	96	73 (16)	74	93	69	246
Ragout2 (multi)	4700 (18)	66 (26)	83	70 (29)	79	95	83	85
AncST pan	800 (14)	62 (60)	83	73 (70)	70	88	78	6
AncST tro	800 (14)	65 (62)	88	75 (72)	70	88	80	7
Multi-AncST	800 (14)	63 (61)	84	73 (70)	71	88	79	6

Table 3 Scaffolding results for the chopped human genome with at least 10% rearranged sequence. For details we refer to the caption of Table 1

Pipeline	Time[min]	NW			adj.			chr. breaks
		prec.	spec.	DNA cov.	prec.	spec.	DNA cov.	
CSAR pan	3300	48 (51)	51	40 (14)	72	75	68	159
CSAR tro	3400	52 (18)	55	43 (16)	73	76	69	172
Multi-CSAR	6800	77 (3)	78	71 (4)	33	56	49	1687
RagTag pan	29	66 (63)	77	67 (63)	76	83	71	23
RagTag tro	27	70 (66)	80	69 (66)	77	83	72	16
Multi-RagTag	58	42 (5)	85	67 (8)	29	73	57	433
ntJoin pan	20	67 (65)	81	69 (66)	78	88	74	9
ntJoin tro	20	70 (67)	84	72 (68)	79	88	74	10
Multi-ntJoin	28	77 (22)	96	69 (16)	73	91	62	243
Ragout2 (multi)	4800 (18)	64 (29)	81	65 (25)	78	95	77	82
AncST pan	850 (15)	63 (60)	83	69 (67)	70	88	74	5
AncST tro	850 (15)	65 (63)	87	72 (69)	71	88	74	4
Multi-AncST	1700 (32)	64 (62)	84	70 (67)	72	88	74	3

In terms of speed, the minimizer-based `RagTag` and `ntJoin` are orders of magnitude faster than the other tools if the extensive alignments needed by the other tools are not precomputed. A more detailed breakdown of the resource consumption of the `AncST` pipeline shows that about a quarter of the compute time is used for the collection of the k-mer statistics with `GenMap` as a basis for determining the anchor candidates. Further two thirds of the effort is expended for the computation of d_0-uniqueness of these candidates through a `blast` comparisons of the anchor candidates against their own genomes. All other tasks in the pipeline account for less than 10% of the wallclock

time. Hence, as can be seen, the computation time for pairwise anchor candidate comparison and scaffolding alone are on par with those of the fastest methods.

Synthetic Data - Evalutaion with `Quast`

Quast evalutaion further supports the results obtained with the metrics described above. The full results and corresponing figures are compiled in Additional File 1. In summary, `CSAR` performs arguably the worst with one exception: It shows the best `auNA` metric for single-reference scaffolding of the chopped but not rearranged version of the human genome. `AncST`, `ntJoin` and `Ragout2` perform relatively well overall. For the multi-reference scenario `AncST` shows robustness while the other tools can usually not only not benefit from using multiple references but performance drops significantly except for `ntJoin` with regards to `auN`.

In Additional File 2 we plot the relative coverage of the 24 human chromosomes (22+`X`+`Y`) estimated by the total length of all `Quast` alignments with the new scaffolds covering this chromosome. It can be seen that while for single reference homology-based scaffolding all tools except `CSAR` produce sensibly large scaffolds, only our pipeline achieves this for multi-reference runs.

Real-Life Data - Evalutaion with `Quast`

As for the synthetic data above, we show `auN`, `auNA` and the number of `misassemblies` as computed by `Quast` to compare the scaffolding results of our pipeline, `ntJoin`, `Ragout2` and `RagTag` shown in Additional File 4. We excluded `Multi-CSAR` from the experiments as it implies resource-intensive whole-genome alignments of all pairs and this tool shows largely disappointing results for the synthetic data set. In Figure 2 we show that the tools are relatively comparable in their results but `AncST` performs best most consistently.

In the following, the detailed analyses are restricted to the two most promising tools based on the contiguity statistics `auN` and `auNA` which are our own and `RagTag`. Firstly, in order to assess whether the multi-reference approach improves over single-reference scenarios, as one would expect from the low quality of the genomes, we compare the fractions of single-reference scaffolds with higher contiguity as measured by `auN`. Comparing all 380 single-reference scaffolds with their respective multi-reference version, in 76 % of cases from `RagTag` the multi-reference version is more contiguous while our pipeline achieves higher contiguity in 90 % of cases. Hence, while both pipelines can utilize the multi-reference approach for a scenario in which a set of lower-quality genomes is available for scaffolding, our pipeline more consistently produces multi-reference scaffolds of higher contiguity. To estimate how many unrealistically large contigs are created in either approach, the number of cases in which the largest contig exceeds 50 million nts, a number significantly higher than any `Drosophila` reference chromosome, are recorded. While no single-reference scaffold's largest contig exceeds 50 milltion nts for either approach, `RagTag` produces 8 and `AncST` 4 multi-reference largest contigs longer than this threshold. Thus, both approaches produce unrealistically large contigs in a significant amount of cases of the multi-reference approach but the issue is more severe for `RagTag`. Thus, in order to estimate the reliability of the contiguity statistics and severity of potential misassemblies we compare the coverage

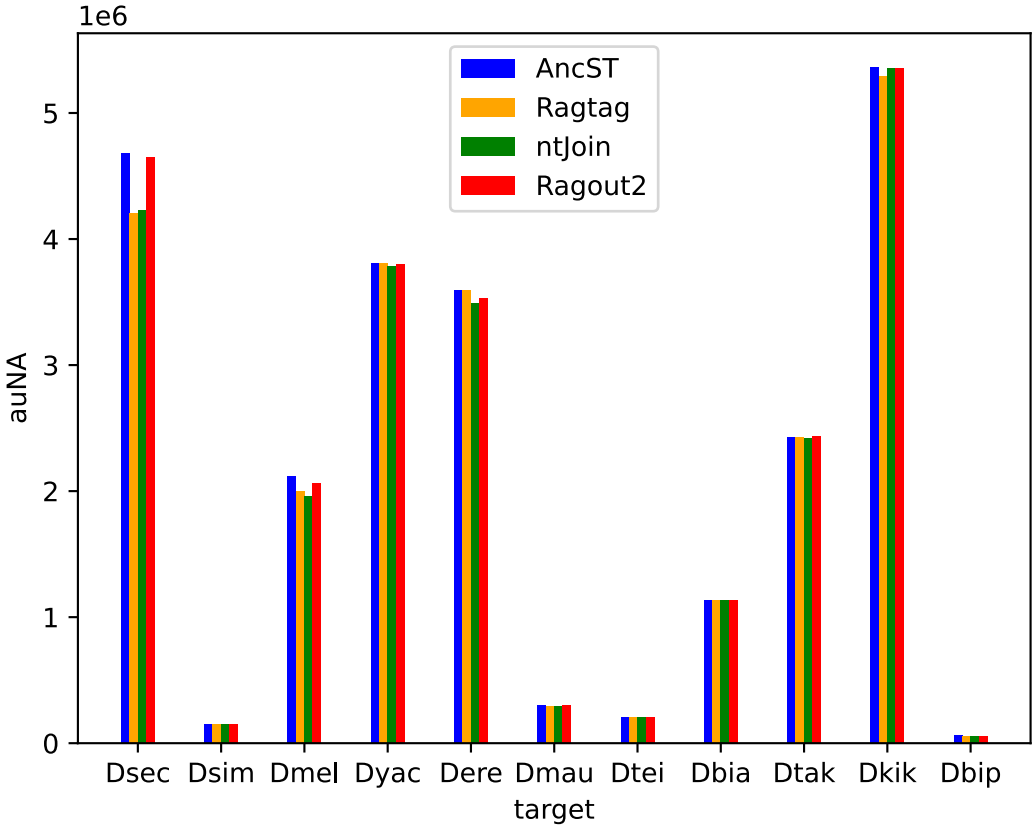

Fig. 2 `auNA` as computed by `Quast` for the 11 `Drosophila` newly scaffolded species with a chromosome- scale reference genome on NCBI. Further details in Table 4.

of large chromosomes of the reference genomes (if available) with potential new scaffolds and single original contigs in some cases. The complete results for each species are displayed in Additional File 5. As an example, see Figure 15 with the extreme case of `Drosophila biarmipes` whose `RagTag` scaffolds display an `auN` value of around 176 million (Fig 13) and a longest contig of length 180 million while the total assembly length is 185 million. Related to the proportion of unrealistically large contigs, `RagTag` scaffolds show more cases in which multiple reference chromosomes are covered by the same scaffold. In order to assess whether `RagTag`'s performance can be improved by using weights for each reference we repeated the scaffolding and analysis using the same weights as in in our pipeline. In essence, using our weights for `RagTag` does not significantly improve its performance (Additional File 6) for this multi-reference scaffolding experiment.

Computational time for scaffolding and merging with RagTag and ntJoin was again lower at around 5 hours and 3 hours, respectively, for this data set using a single core. Computation of AncST anchors took around 11 hours from upload of the accessions to the webserver at http://anchored.bioinf.uni-leipzig.de:8080/ and subsequent scaffolding and merging took less than an hour parallelized at one core used for each species.

6 Conclusions

Homology-based scaffolding is likely to gain increasing importance in non-model-organism genomics in response to the rapidly growing number of high quality genomes that are suitable as references. The phylogenetic distance of available reference genomes to a newly sequenced genome thus shrinks, increasing the accuracy of the reference-based approach and making multi-reference scaffolding more attractive. Hence, the methodology has the potential to provide biologists with large numbers of accessible (close to) chromosome-level genome assemblies for functional analyses, comparative genomics, and evolutionary studies at moderate experimental and computational costs. This is of particular relevance for the large-scale efforts mentioned in the introduction. We have shown here that pre-computed anchor candidates, i.e., sufficiently unique DNA in a genome, can be used effectively as a basis for efficient homology-based contig scaffolding. A pipeline based on our recently released anchor-based synteny detector AncST performs acceptably well in comparison with four established tools. For the multi-reference runs the AncST-based version arguably outperforms the other pipelines, particularly for the real-life data set. Moreover, it seems that our scaffolder yields more biologically realistic and useful results, at least on the test data evaluated here. As previously described, contiguity metrics are too simplistic to properly capture assembly quality [22, 23, 24, 25]. Accordingly, for the synthetic data our pipeline produces scaffolds which cover significant proportions of the respective human reference chromosomes. Furthermore, our pipeline shows robustness in multi-reference runs. In contrast, the other tools produce comparably small and in part significantly worse scaffolds when using the two references. Similarly, AncST also creates significantly more realistic chromosomes in the analysis with real data than the arguably second best performing tool for this task RagTag whose pseudo-chromosomes often cover multiple reference chromosomes. Such misassembly errors eventually will have to be dealt with manually in time-consuming steps and if not detected, can lead to a series of misjudgements in any comparative genomics analysis.

Since the performance bottlenecks are external programs, in particular MUMmer [26], GenMap [27] and minimap2 [7], cactus and blast, we did not benchmark memory consumption in detail. The choice of these tools is partly reflected in the performance statistics: CSAR uses MUMmer to create a complete genome alignment and Ragout2 is based on multiple sequence alignments from cactus, at least for genomes as large as in this study. Thus, these tools theoretically have the the best resolution while also consuming the most resources. In this study it can also be observed that whole-genome and multiple sequence alignments do in fact not always yield best or even relatively good performance. AncST uses a k-mer heuristic based on GenMap k-mer counts and

then uses `blast` as an index-based sequence alignment tool both to infer potentially good anchor candidates per genome and to find hits between the sets of candidates. By largely excluding duplicated regions of the genome, the order of these high-confidence alignments can then effectively be used for contig scaffolding. Our strategy hence consumes less resources than a full genome alignment. Nevertheless, it reproduces most of the useful signal and arguably leaves out some of the conflicting signal from duplicated sequences which may negatively impact the tools based on whole-genome and multiple alignments. `RagTag` also works with alignments but produced by the much faster `minimap` which itself is based on minimizer sketches. Such sketches are used in `ntJoin` directly to scaffold assemblies.

The current version of the `AncST`-based pipeline is not intended for heavy-duty production but rather serves as a proof-of-concept for the practical utility of the anchor-based approach. As discussed in [12], it may be possible to replace `GenMap` by alternative statistics of sequence-uniqueness, in particular variants of the Lempel-Ziv complexity, see e.g. [28]. Moreover, it is likely possible to adapt minimizer sketches as filters in the process of comparing anchor candidates between genomes. In this manner, both of the present performance bottlenecks in the `AncST` software could be addressed.

A distinctive advantage of the anchor-based approach is that anchor candidates can be precomputed and stored as simple annotation files that represent anchors as coordinate pairs associated with their level of the interval's d_0-uniqueness. Pairwise comparisons are then restricted to these candidates, and by construction, result in unambiguous alignments. In fact, disregarding this preprocessing of the genomes reduces the computation times to figures similar to the fastest tools, while asserting a more robust contig ordering.

Non-colinear anchor alignments yield strong evidence for breakpoints arising from genome rearrangements. The anchor matches themselves, moreover, can be fed directly into the analysis of macrosynteny, which is usually performed as part of genome assembly projects. For instance, we can readily track possible inversion scenarios by enumerating regions for which `AncST` anchors align on the opposing strand in considerable numbers. Translocations are indicated by significant colinear stretches of alignments with a breakpoint in one of the species. The current implementation of the `AncST`-based pipeline identifies contigs with alignments to more than one reference chromosome and those with alignments in opposite reading directions on the same reference chromosome. These are strong indicators for rearrangements relative to the reference genome. In principle, these data could be used to extend scaffolds beyond such breakpoints. Whether this can be done with acceptable reliability, however, is currently subject to investigation.

Moreover, given a case where it is advisable to use multiple references for contig scaffolding, our approach seems to perform well and is scalable since `AncST` itself is scalable due the fact that its resource consumption is dominated by the preprocessing that is performed independently for each genome. In contrast, `MUMmer`, `minimap` and `cactus` are genuine pairwise aligners and thus require full computational efforts for all pairs of genomes under consideration. As we have seen, the near-uniqueness of anchor-based alignments adds to the performance particularly in a multiple reference scenario, without incurring significant extra computational costs.

Availability and requirements

Project name: AncST Scaffolder
Project home page: github.com/Norsbus/AncST
Operating system(s): Linux
Programming language: Python
Other requirements: none
License: GPL-3.0
Any restrictions to use by non-academics: none

7 Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

The supplementary material is available at Zenodo (https://zenodo.org/records/17113483). The archive includes all used and created genomes as well as the scripts used to create the latter. Furthermore, all rearrangements are documented and precomputed `AncST` anchors are provided. The code used for the computational experiments and evaluation are available in the supplementary material including the version of the scaffolder used for this paper.

Competing interests

The authors have no competing interests.

Funding

This work was funded by the German Research Foundation as part of SPP 2349 *"Genomic Basis of Evolutionary Innovations (GEvol)"* (grant no. 502862570). PFS acknowledges the financial support by the Federal Ministry of Education and Research of Germany (BMBF) through DAAD project 57616814 (SECAI, School of Embedded Composite AI), and jointly with the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research *Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig*, project identification number: SCADS24B.

Authors' contributions

All authors conceived the study. KKK implemented the ideas, conducted the computational experiments and wrote the first draft of the manuscript. SL tested the pipeline. All authors contributed to editing the manuscript.

Acknowledgments

Not applicable

References

[1] Luo J, Wei Y, Lyu M, Wu Z, Liu X, Luo H, et al. A comprehensive review of scaffolding methods in genome assembly. Briefings in Bioinformatics. 2021;22:bbab033. https://doi.org/10.1093/bib/bbab033.

[2] Burton JN, Adey A, Patwardhan RP, Qiu R, Kitzman JO, Shendure J. Chromosome-scale scaffolding of *de novo* genome assemblies based on chromatin interactions. Nat Biotechnol. 2013;31(12):1119–1125. https://doi.org/10.1038/nbt.2727.

[3] Yamaguchi K, Kadota M, Nishimura O, Ohishi Y, Naito Y, Kuraku S. Technical considerations in Hi-C scaffolding and evaluation of chromosome-scale genome assemblies. Mol Ecol. 2021;30(23):5923–5934. https://doi.org/10.1111/mec.16146.

[4] Chen KT, Liu CL, Huang SH, Shen HT, Shieh YK, Chiu HT, et al. CSAR: a contig scaffolding tool using algebraic rearrangements. Bioinformatics. 2018;34:109–111. https://doi.org/10.1093/bioinformatics/btx543.

[5] Alonge M, Soyk S, Ramakrishnan S, Wang X, Goodwin S, Sedlazeck FJ, et al. RaGOO: fast and accurate reference-guided scaffolding of draft genomes. Genome Biology. 2019;20:224. https://doi.org/10.1186/s13059-019-1829-6.

[6] Coombe L, Nikolić V, Chu J, Birol I, Warren RL. ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. Bioinformatics. 2020;36(12):3885–3887. https://doi.org/10.1093/bioinformatics/btaa253.

[7] Li H. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018;34(18):3094–3100. https://doi.org/10.1093/bioinformatics/bty191.

[8] Kolmogorov M, Armstrong J, Raney BJ, Streeter I, Dunn M, Yang F, et al. Chromosome assembly of large and complex genomes using multiple references. Genome Res. 2018;28(11):1720–1732. https://doi.org/10.1101/gr.236273.118.

[9] Alonge M, Lebeigle L, Kirsche M, Jenike K, Ou S, Aganezov S, et al. Automated assembly scaffolding using RagTag elevates a new tomato system for high-throughput genome editing. Genome biology. 2022;23:258. https://doi.org/10.1186/s13059-022-02823-7.

[10] Chen KT, Shen HT, Lu CL. Multi-CSAR: a multiple reference-based contig scaffolder using algebraic rearrangements. BMC Systems Biology. 2018;12:69–75. https://doi.org/10.1186/s12918-018-0654-y.

[11] Lischer HE, Shimizu KK. Reference-guided de novo assembly approach improves genome reconstruction for related species. BMC bioinformatics. 2017;18(1):474. https://doi.org/10.1186/s12859-017-1911-6.

[12] Käther KK, Remmel A, Lemke S, Stadler PF. Unbiased anchors for reliable genome-wide synteny detection. Alg Mol Biol. 2025;20:5. https://doi.org/10.1186/s13015-025-00275-9.

[13] Schrinner S, Goel M, Wulfert M, Spohr P, Schneeberger K, Klau GW. Using the longest run subsequence problem within homology-based scaffolding. Algorithms

for Molecular Biology. 2021;16:11. https://doi.org/10.1186/s13015-021-00191-8.
[14] Ghiurcuta CG, Moret BME. Evaluating synteny for improved comparative studies. Bioinformatics. 2014;30(12):i9–i18. https://doi.org/10.1093/bioinformatics/btu259.
[15] Käther K, Lemke S, Stadler PF. Annotation-free Identification of Potential Synteny Anchors. In: Rojas I, Valenzuela O, Rojas Ruiz F, Herrera LJ, Ortuño F, editors. International Work-Conference on Bioinformatics and Biomedical Engineering (IWWBIO'23). vol. 13919 of Lect. Notes Comp. Sci. Cham: Springer; 2023. p. 217–230.
[16] Edmonds J. Path, trees, and Flowers. Canadian J Math. 1965;17:449–467. https://doi.org/10.4153/CJM-1965-045-4.
[17] Kolmogorov V. Blossom V: a new implementation of a minimum cost perfect matching algorithm. Mathematical Programming Computation. 2009;1:43–67. https://doi.org/10.1007/s12532-009-0002-8.
[18] Li K, Xu P, Wang J, Yi X, Jiao Y. Identification of errors in draft genome assemblies at single-nucleotide resolution for quality assessment and improvement. Nature Communications. 2023;14(1):6556. https://doi.org/10.1038/s41467-023-42336-w.
[19] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48(3):443–453. https://doi.org/10.1016/0022-2836(70)90057-4.
[20] Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics. 2013;29(8):1072–1075. https://doi.org/10.1093/bioinformatics/btt086.
[21] Li H. auN: a New Metric to Measure Assembly Contiguity; 2020. Available from: https://lh3.github.io/2020/04/08/a-new-metric-on-assembly-contiguity.
[22] Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, et al. GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome research. 2012;22(3):557–567. https://doi.org/10.1101/gr.131383.111.
[23] Thrash A, Hoffmann F, Perkins A. Toward a more holistic method of genome assembly assessment. BMC bioinformatics. 2020;21(Suppl 4):249. https://doi.org/10.1186/s12859-020-3382-4.
[24] Wang P, Wang F. A proposed metric set for evaluation of genome assembly quality. Trends in Genetics. 2023;39(3):175–186. https://doi.org/10.1016/j.tig.2022.10.005.
[25] Narzisi G, Mishra B. Comparing de novo genome assembly: the long and short of it. PloS one. 2011;6(4):e19175. https://doi.org/10.1371/journal.pone.0019175.
[26] Marçais G, Delcher AL, Phillippy AM, Coston R, Salzberg SL, Zimin A. MUMmer4: A fast and versatile genome alignment system. PLoS Comp Biol. 2018;14(1):e1005944. https://doi.org/10.1371/journal.pcbi.1005944.
[27] Pockrandt C, Alzamel M, Iliopoulos CS, Reinert K. GenMap: ultra-fast computation of genome mappability. Bioinformatics. 2020;36(12):3687–3692. https://doi.org/10.1093/bioinformatics/btaa222.
[28] Pirogov A, Pfaffelhuber P, Börsch-Haubold A, Haubold B. High-complexity

regions in mammalian genomes are enriched for developmental genes. Bioinformatics. 2019;35(11):1813–1819. https://doi.org/10.1093/bioinformatics/bty922.

Additional File 1

auN(A) and Misassemblies as computed by Quast

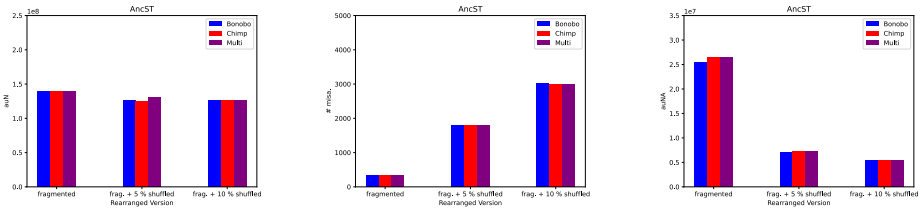

Fig. 3 (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `AncST`.

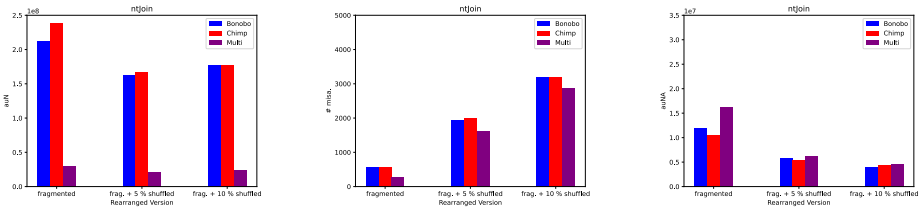

Fig. 4 (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `ntJoin`.

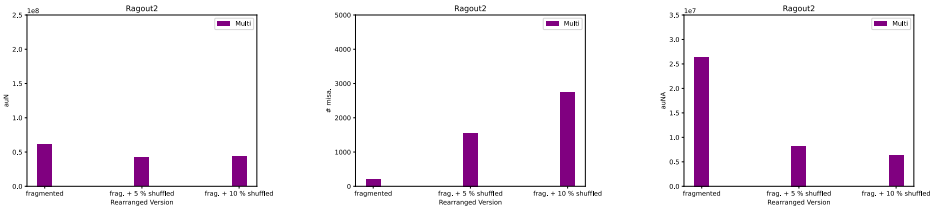

Fig. 5 (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `Ragout2`.

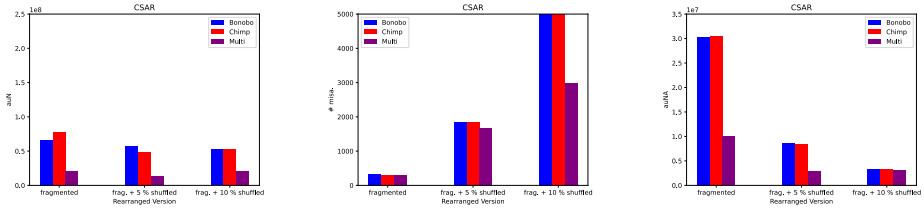

Fig. 6 (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `CSAR`.

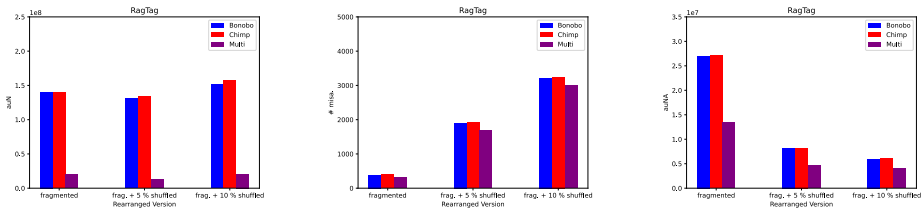

Fig. 7 (Left:) *auN*, (Middle:) *misassemblies* and (Right:) *auNA* as computed by `Quast` for the new scaffolds produced by `RagTag`.

Additional File 2

Coverage of Main Human Chromosomes with Quast Alignments of New Scaffolds

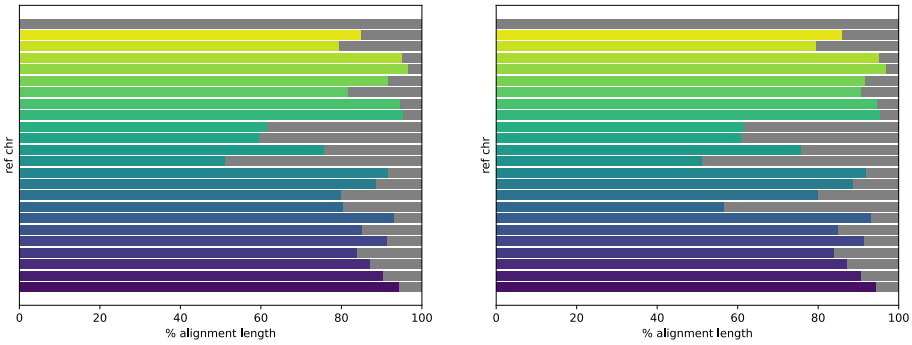

Fig. 8 Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `AncST`. Relative coverage (X-axis) of 24 human chromosomes (22+`X`+`Y`) (Y-axis) by all alignments produced by `Quast`. First, the total length of all alignments of a reference chromosome with any new scaffolds is noted. Then each new scaffold is assigned its relative coverage as the proportion of the length of its alignments with the reference. Scaffolds covering at least 50 % of a reference are colored and the rest is gray. Relative coverage by one contig is bordered by white vertical lines.

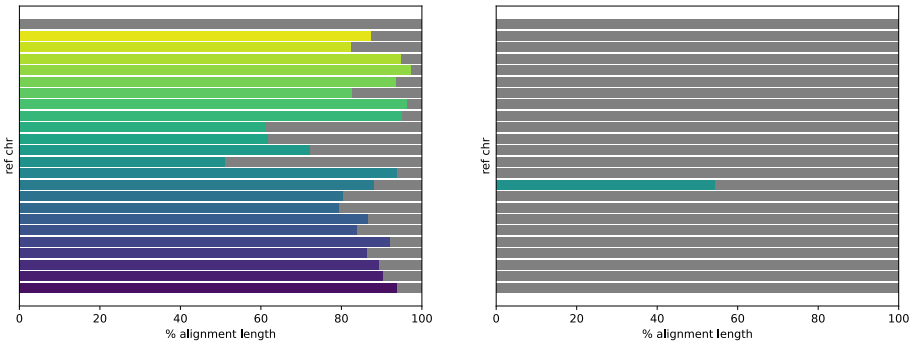

Fig. 9 Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `ntJoin`. Details can be found in caption of 8.

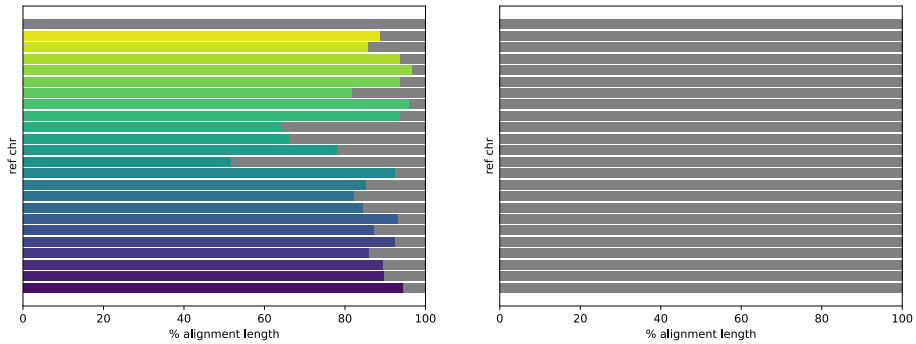

Fig. 10 Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `RagTag`. Details can be found in caption of 8.

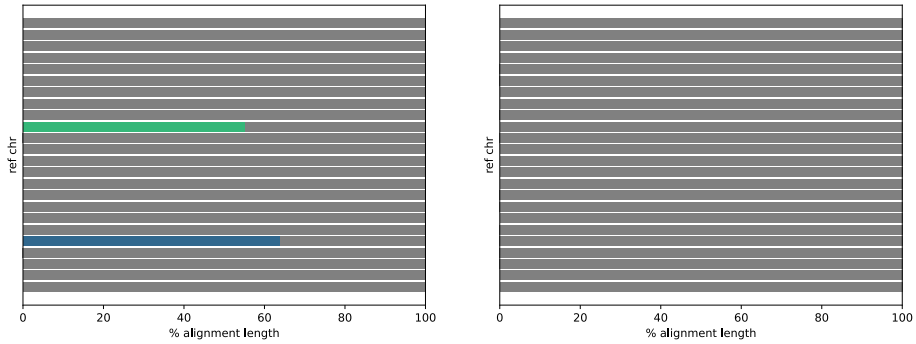

Fig. 11 Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against (left:) chimp and (right:) chimp and bonobo from `CSAR`. Details can be found in caption of 8.

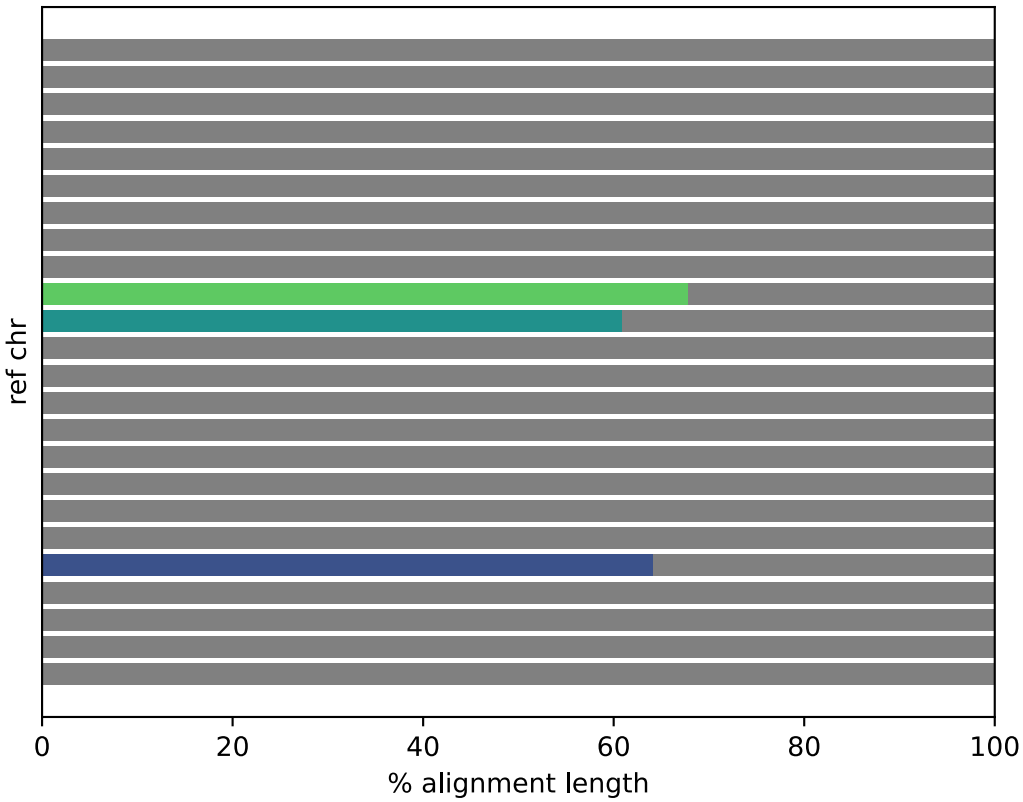

Fig. 12 Shown are data based on the results of scaffolding the human genome with at least 5 % rearranged sequence against chimp and bonobo from `Ragout2`. Details can be found in caption of 8.

Additional File 3

Drosophila Assemblies Used

Table 4 Genomes of `Drosophila` species used. `same` in `NCBI Ref.`(erence) means that the Contig or Scaffold level assembly used as a scaffolding target is also marked as the reference chromosome for this species on NCBI.

Accession	Species	Abbreviation	Assembly Level	NCBI Ref.	Identifier
GCA_018904445.1	`D. sechellia`	Dsec	Scaffold	GCF_004382195.2	A
GCA_039725655.1	`D. simulans`	Dsim	Contig	GCF_016746395.2	B
GCA_000778455.1	`D. melanogaster`	Dmel	Contig	GCF_000001215.4	C
GCA_018904385.1	`D. yakuba`	Dyac	Contig	GCF_016746365.2	D
GCA_018904525.1	`D. erecta`	Dere	Scaffold	GCF_003286155.1	E
GCA_018904475.1	`D. mauritiana`	Dmau	Scaffold	GCF_004382145.1	F
GCA_018903625.1	`D. teissieri`	Dtei	Contig	GCF_016746235.2	G
GCA_005876975.1	`D. orena`	Dore	Contig	same	H
GCF_018153835.1	`D. eugracilis`	Deug	Contig	same	I
GCA_018148935.1	`D. biarmipes`	Dbia	Contig	GCF_025231255.1	J
GCA_018152695.1	`D. takahashii`	Dtak	Contig	GCF_030179915.1	K
GCF_018152265.1	`D. ficusphila`	Dfic	Contig	same	L
GCF_018152505.1	`D. elegans`	Dele	Contig	same	M
GCF_018152115.1	`D. rhopaloa`	Drho	Contig	same	N
GCA_008042655.1	`D. burlai`	Dbur	Scaffold	same	O
GCA_018152535.1	`D. kikkawai`	Dkik	Contig	GCF_030179895.1	P
GCA_008042735.1	`D. leontia`	Dleo	Scaffold	same	Q
GCA_021223765.1	`D. bipectinata`	Dbip	Scaffold	GCF_030179905.1	R
GCA_018153235.1	`D. malerkotliana`	Dmal	Contig	same	S
GCA_018148915.1	`D. ananassae`	Dana	Contig	same	T

Additional File 4

Scaffolding Evaluation of Drosophilas

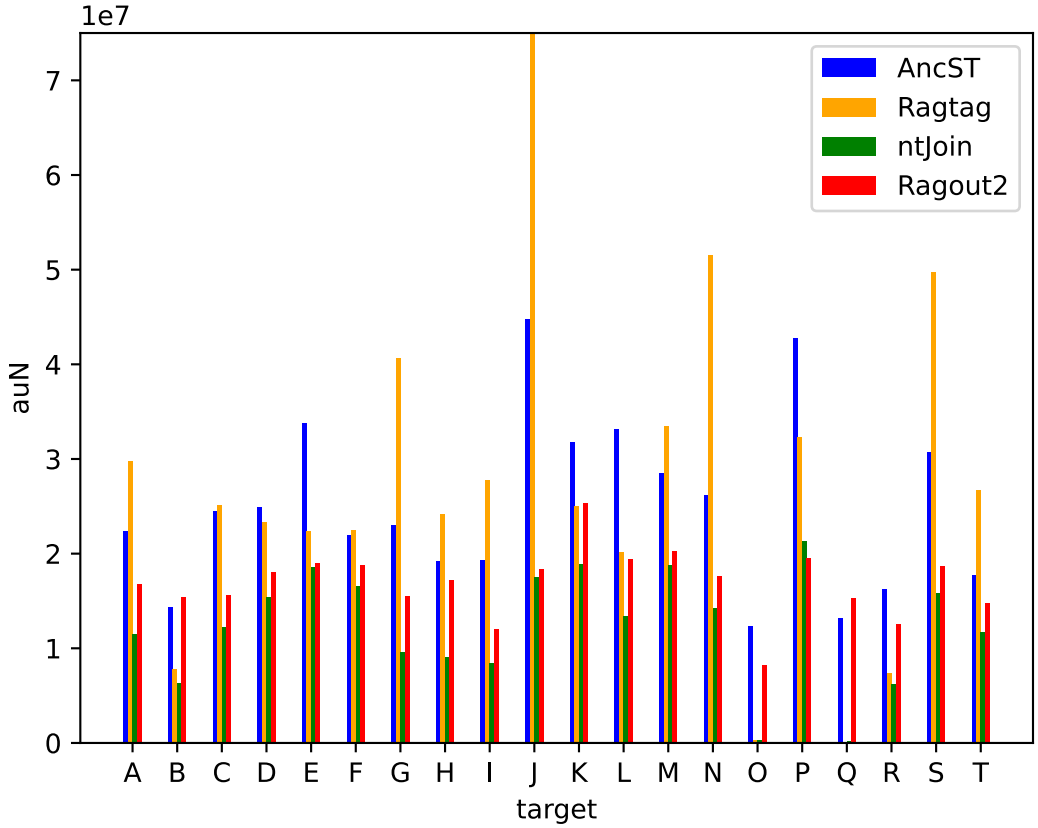

Fig. 13 *auN* as computed by `Quast` for 20 `Drosophila` newly scaffolded species. The new scaffolds for computed by `RagTag` for `Drosophila biarmipes` (J) show an *auN* of around 180 million while we set the upper limit of the y-axis to 75 million for clearer display. Identifier correspondence and further details in Table 4.

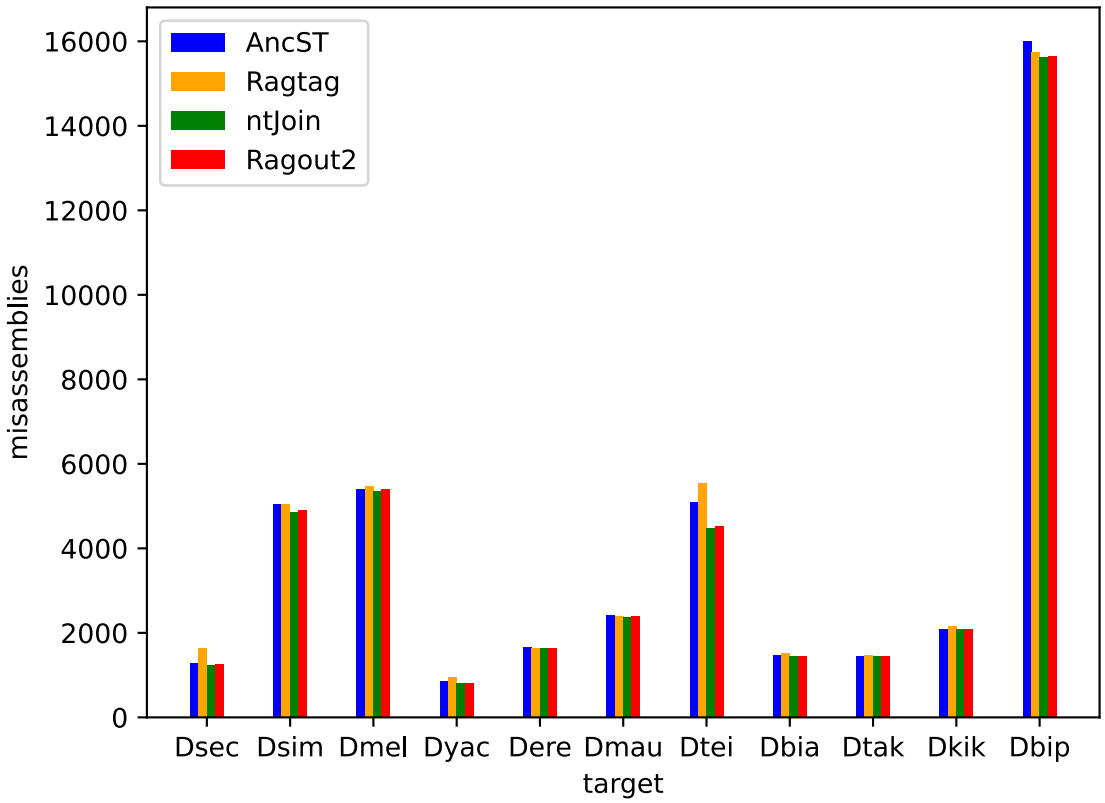

Fig. 14 Number if *misassemblies* as computed by `Quast` for the 11 `Drosophila` newly scaffolded species with a chromosome- scale reference genome on NCBI. Further details in Table 4.

Additional File 5

Assessment of Reference Chromosome Coverage with New Scaffolds

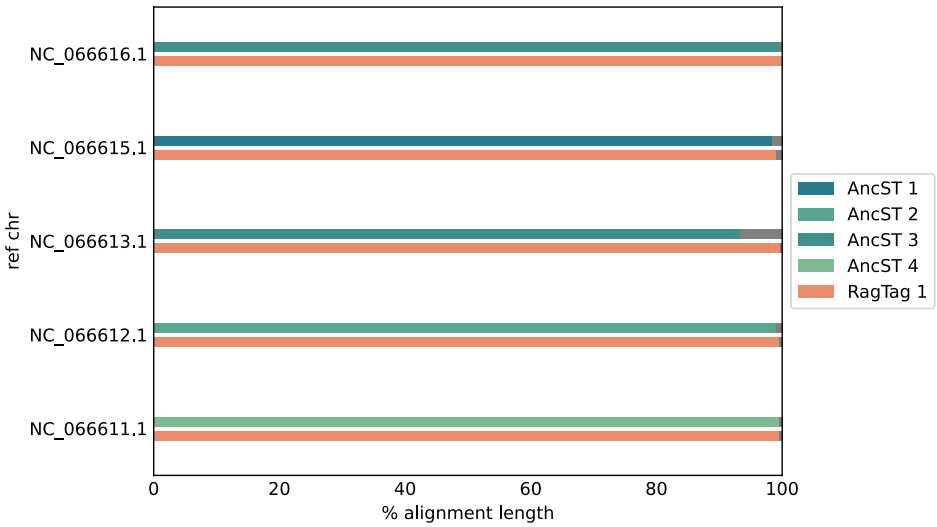

Fig. 15 Shown are all reference chromosomes for the `Drosophila biarmipes` official reference assembly on NCBI on the y-axis. For each reference chromosome, the upper bar displays results computed with the `AncST`-based pipeline and the lower bar the ones from `RagTag`. The bars are stacked according to the coverage of each reference chromosome by new scaffolds from the respective tool. The coverage is estimated by the total alignment length of all `minimap` alignments recorded in the output of `Quast`. Only contigs/scaffolds covering at least a third of the total alignment length are drawn colored while the rest is kept gray. Each color represents a different new scaffold which are indicated in the legend.

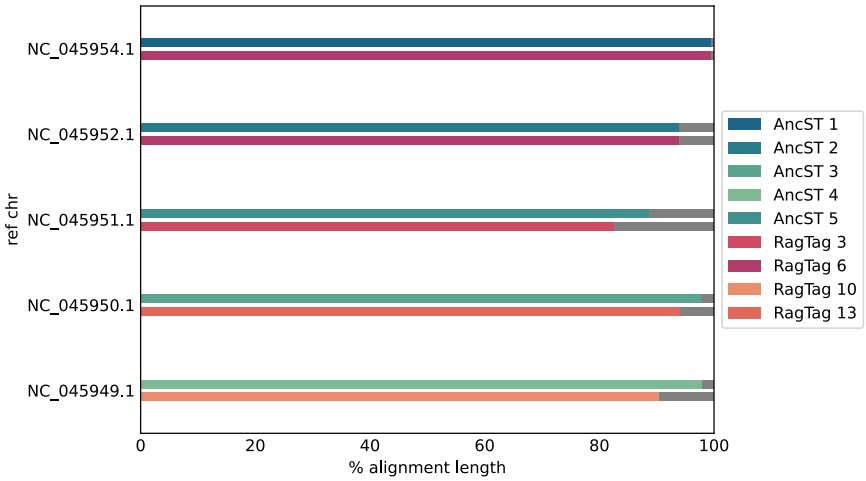

Fig. 16 Assembly: GCA_018904445.1. Shown are all reference chromosomes for a species' official reference assembly on NCBI on the y-axis. For each reference chromosome, the upper bar displays results computed with the `AncST`-based pipeline and the lower bar the ones from `RagTag`. The bars are stacked according to the coverage of each reference chromosome by original contigs or new scaffolds from the respective tool. Each color represents a different original contig or new scaffold which are indicated in the legend. Only contigs/scaffolds covering at least a third of the total alignment length are drawn colored while the rest is kept gray. The coverage is estimated by the total alignment length of all `minimap` alignments recorded in the output of `Quast`. `From A.` means as computed by `AncST` and `From R.` means as computed by `RagTag`.

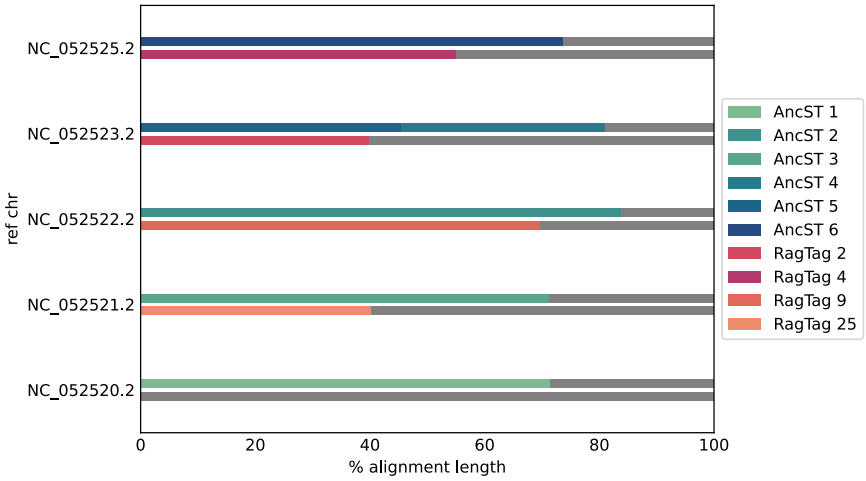

Fig. 17 Assembly: GCA_039725655.1. See caption of Figure 16.

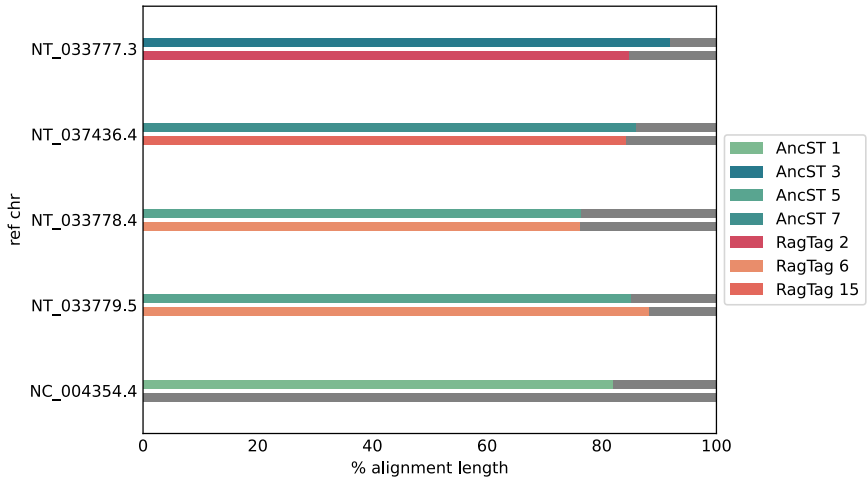

Fig. 18 Assembly: GCA_000778455.1. See caption of Figure 16.

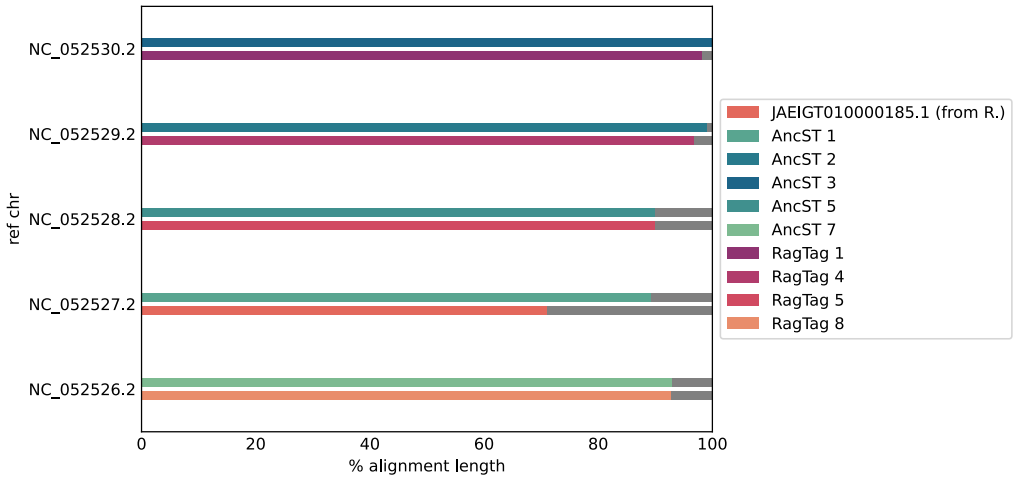

Fig. 19 Assembly: GCA_018904385.1. See caption of Figure 16.

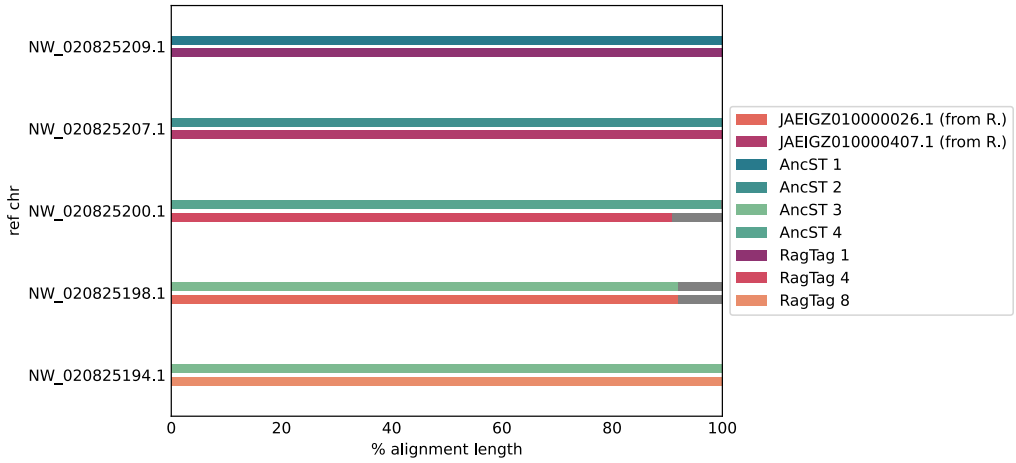

Fig. 20 Assembly: GCA_018904525.1. See caption of Figure 16.

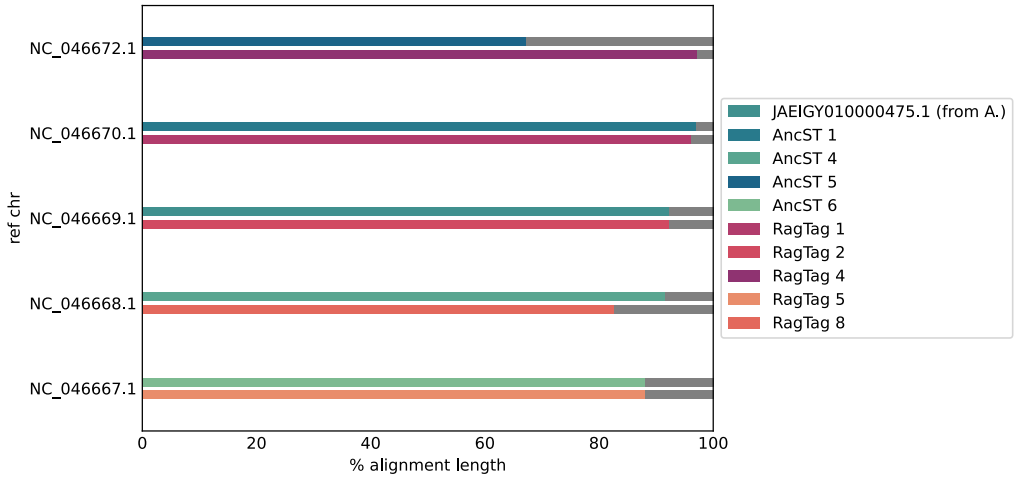

Fig. 21 Assembly: GCA_018904475.1. See caption of Figure 16.

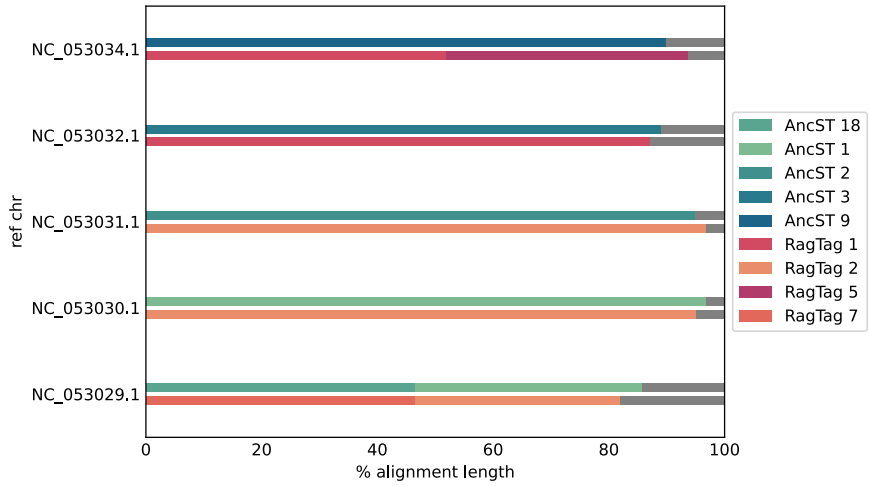

Fig. 22 Assembly: GCA_018903625.1. See caption of Figure 16.

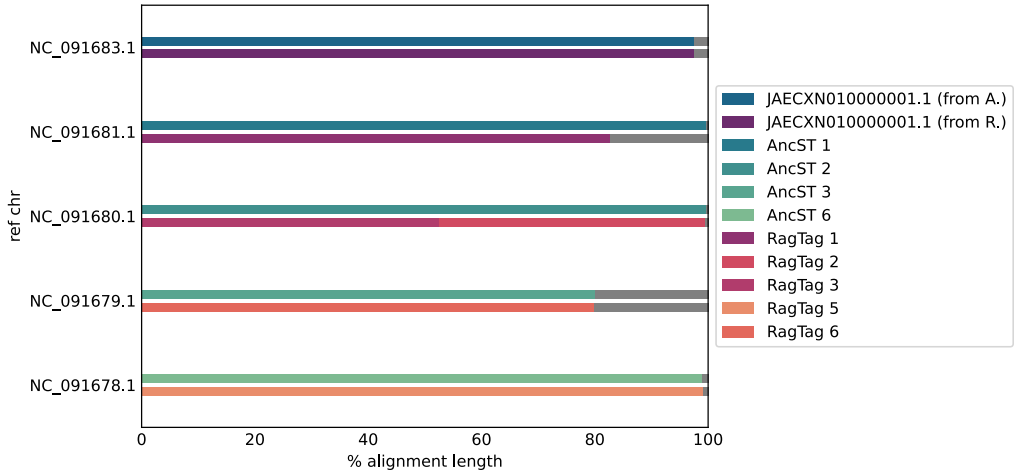

Fig. 23 Assembly: GCA_018152695.1. See caption of Figure 16.

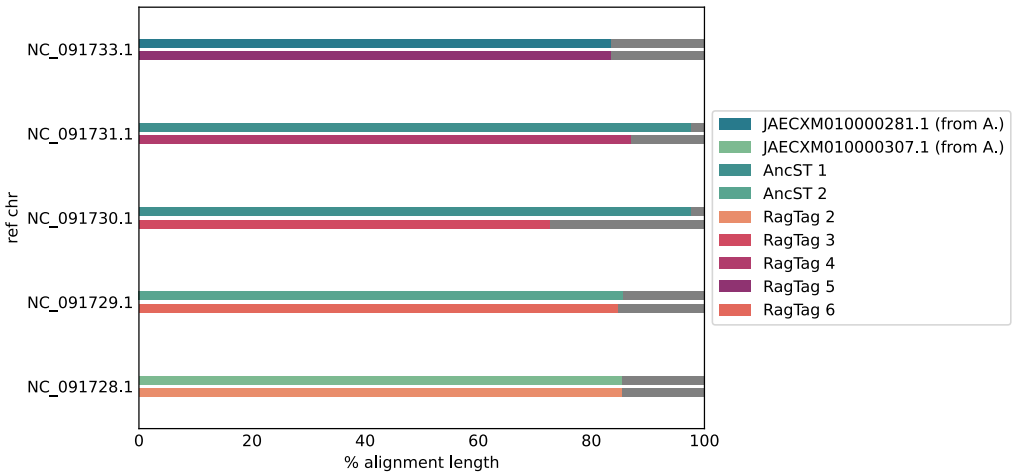

Fig. 24 Assembly: GCA_018152535.1. See caption of Figure 16.

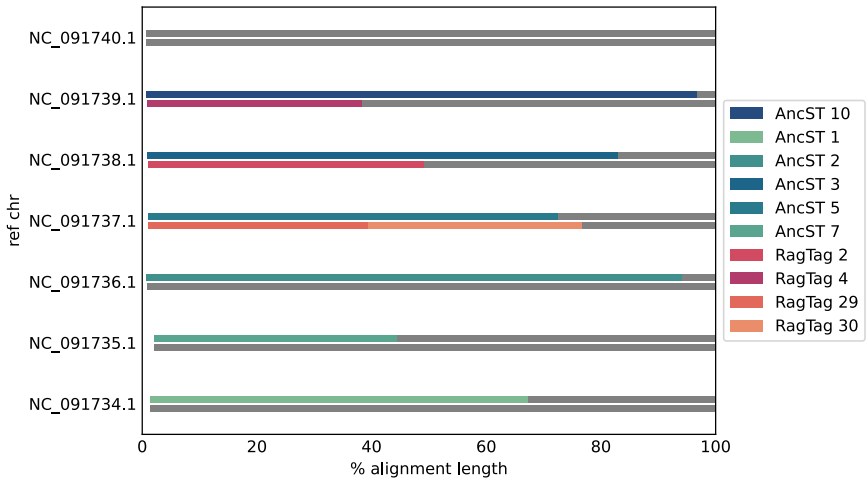

Fig. 25 Assembly: GCA_021223765.1. See caption of Figure 16.

Additional File 6

Influence of Using Weights for RagTag and ntJoin on auN and Number of Misassemblies for Drosophila Scaffolds

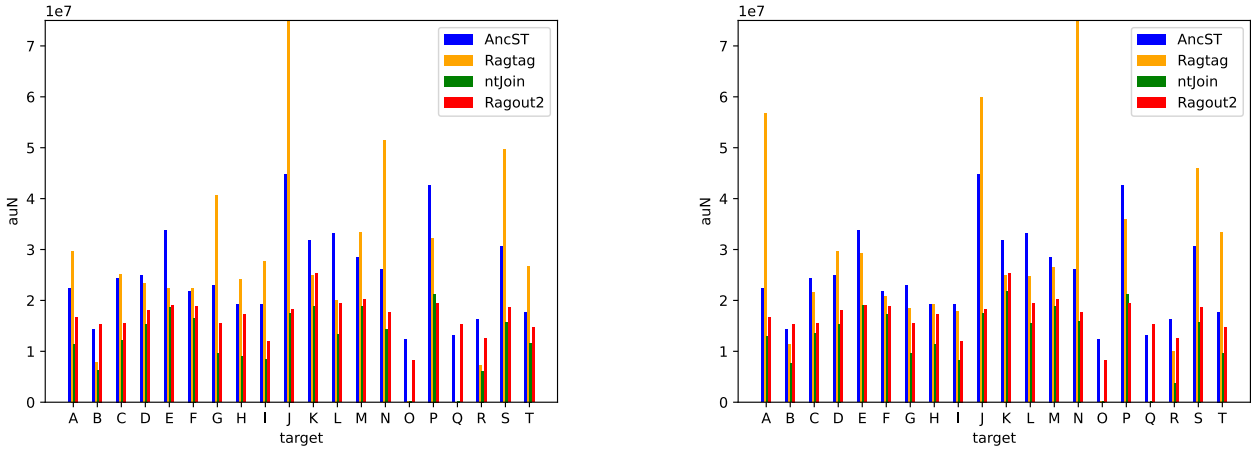

Fig. 26 Left: Same plot as Fig. 13. Right: Same analysis but using `AncST` weights for `RagTag` and `ntJoin`.

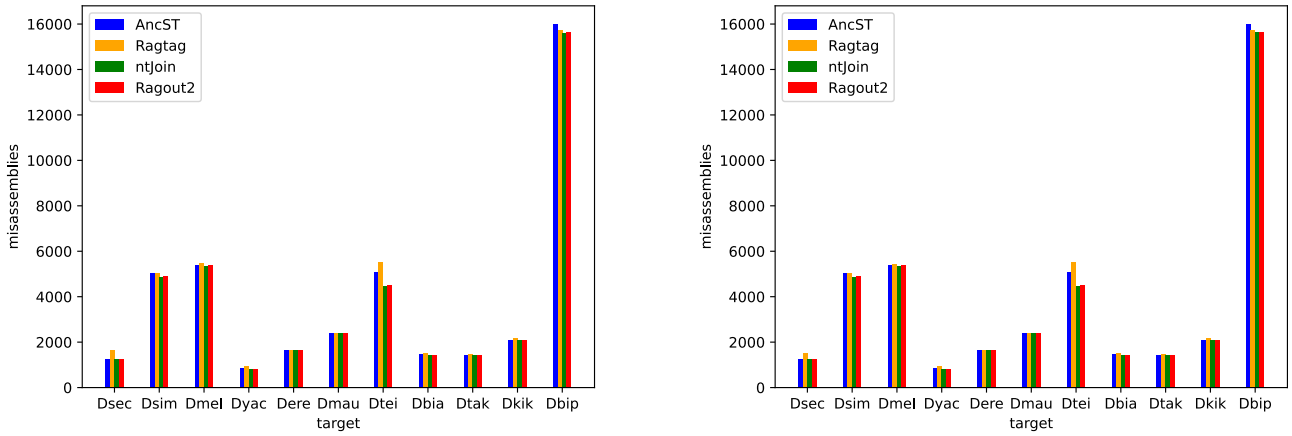

Fig. 27 Left: Same plot as Fig. 14. Right: Same analysis but using `AncST` weights for `RagTag` and `ntJoin`.
(yes, the two plots are different)

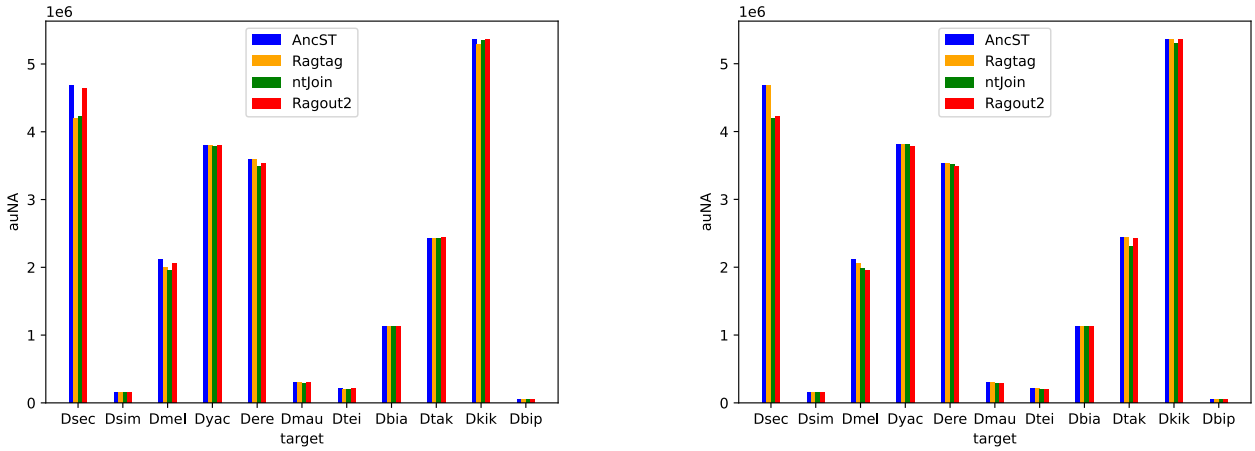

Fig. 28 Left: Same plot as Fig. 2. Right: Same analysis but using `AncST` weights for `RagTag` and `ntJoin`.