Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

HHS Public Access

Author manuscript

Nature. Author manuscript; available in PMC 2021 May 11.

Published in final edited form as:

Nature. 2020 November ; 587(7833): 246–251. doi:10.1038/s41586-020-2871-y.

Progressive Cactus: a multiple-genome aligner for the thousand-genome era

Joel Armstrong[1], **Glenn Hickey**[1], **Mark Diekhans**[1], **Ian T. Fiddes**[1], **Adam M. Novak**[1], **Alden Deran**[1], **Qi Fang**[2,3,4], **Duo Xie**[2,3,5], **Shaohong Feng**[2,3,6], **Josefin Stiller**[4], **Diane Genereux**[7], **Jeremy Johnson**[7], **Voichita Dana Marinescu**[8], **Jessica Alföldi**[7], **Robert S. Harris**[9], **Kerstin Lindblad-Toh**[7,8], **David Haussler**[1,10], **Elinor Karlsson**[7,11,12], **Erich D. Jarvis**[10,13], **Guojie Zhang**[*,2,4,6,14], **Benedict Paten**[†,1]

[1]UC Santa Cruz Genomics Institute, UC Santa Cruz, Santa Cruz, CA 95060, USA

[2]China National GeneBank, BGI–Shenzhen, Jinsha Road, Shenzhen 518120, China

[3]BGI-Shenzhen, Beishan Industrial Zone, Shenzhen 518083, China

[4]Section for Ecology and Evolution, Department of Biology, University of Copenhagen, DK-2100 Copenhagen, Denmark

[5]University of Chinese Academy of Sciences, Beijing 100049, China

[6]State Key Laboratory of Genetic Resources and Evolution, Kunming Institute of Zoology, Chinese Academy of Sciences, Kunming 650223, China

[7]Broad Institute of Harvard and Massachusetts Institute of Technology (MIT), 7 Cambridge Center, Cambridge, Massachusetts 02142, USA

[8]Science for Life Laboratory, Department of Medical Biochemistry and Microbiology, Uppsala University, Box 582, SE-751 23 Uppsala, Sweden

[9]Department of Biology, The Pennsylvania State University, 508 Wartik Labs, University Park, PA 16802, USA

[10]Howard Hughes Medical Institute, Chevy Chase, MD, USA

[11]Program in Molecular Medicine, University of Massachusetts Medical School, Worcester, MA 01655, USA

[12]Bioinformatics and Integrative Biology, University of Massachusetts Medical School, Worcester, MA 01655, USA

Users may view, print, copy, and download text and data-mine the content in such documents, for the purposes of academic research, subject always to the full Conditions of use:http://www.nature.com/authors/editorial_policies/license.html#terms

[*]Corresponding author. guojie.zhang@bio.ku.dk. [†] Corresponding author. bpaten@ucsc.edu.

Author Contributions

JA, GH, DH, MD, AMN, ITF, AD, RSH, and BP developed Progressive Cactus and key software upon which it depends. JA, QF, DX, SF, and GS created and analyzed the avian sub-alignment. JA, DG, JJ, and VDM created and analyzed the mammalian sub-alignment. DH, KL, EK, EJ, GZ, and BP directed and developed the work. MD and JA performed the human-chicken alignment comparisons. JA performed the remaining analyses. JA and BP wrote the manuscript and all authors edited it.

Competing Interests Statement

The authors have no competing interests to declare.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

[13]Laboratory of Neurogenetics of Language, The Rockefeller University, NY, USA

[14]Center for Excellence in Animal Evolution and Genetics, Chinese Academy of Sciences, Kunming, 650223, China

Abstract

New genome assemblies have been arriving at a rapidly increasing pace, thanks to decreases in sequencing costs and improvements in third-generation sequencing technologies [1, 2, 3]. For example, the number of vertebrate genome assemblies currently in the NCBI database [4] increased by over 50% to 1485 assemblies in the year to July 2019. In addition to this influx of assemblies from different species, new human *de novo* assemblies [5] are being produced, which enable analysis of not just small polymorphisms, but also complex, large-scale structural differences between human individuals and haplotypes. This coming era and its unprecedented amount of data offer the opportunity to unlock many insights into genome evolution but also presents challenges in adapting our analysis methods to meet the increased scale. Cactus[6], a reference-free multiple genome alignment program, has been shown to be highly accurate, but the existing implementation scales poorly with increasing numbers of genomes, and struggles in regions of highly duplicated sequence. We describe progressive extensions to Cactus to create Progressive Cactus. Progressive Cactus enables the reference-free alignment of tens to thousands of large vertebrate genomes while maintaining high alignment quality. We describe results from an alignment of over 600 amniote genomes, which is to our knowledge the largest multiple vertebrate genome alignment yet created.

Comparative genomics analyses, including species-tree inference [7, 8], comparative annotation [9, 10], and selection detection [11, 12], require genome alignments. Multi-species genome alignment involves creating a mapping from each region of each genome to a corresponding region in each other genome, taking into account the possibility of complex rearrangements and copy number changes [13]. Genome aligners are one of the most fundamental tools used in comparative genomics, but since the problem is difficult, different aligners frequently give different results [14], and many intentionally limit the alignments they produce to simplify the problem. Two of the most common limitations are *reference-bias*, the result of constraining a multiple alignment to only regions present in a single reference genome, and restricting the alignment to be *single-copy*, which allows only a single alignment in any column in any given genome, causing the alignment to miss multiple-orthology relationships created by lineage-specific duplications. Cactus [6] is a genome alignment program that has neither of these restrictions; it is capable of generating a reference-free multiple alignment that allows detecting multiple-orthology relationships.

The version of Cactus available in 2012 performed very well in the Alignathon [14], an evaluation of genome aligners. However, the runtime of that initial iteration of Cactus scaled quadratically with the total number of bases in the alignment problem, making alignment of more than about ten vertebrate genomes completely impractical. To address these difficulties, we present fundamental changes to the Cactus process that incorporate a progressive alignment strategy [15], which changes the runtime of the alignment to scale linearly with the number of genomes. We show that the result, which we call Progressive

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Cactus, is an aligner that retains state-of-the-art accuracy, and continues to lack reference bias, but which is tractable to use on hundreds to thousands of large, vertebrate-sized input genomes. Progressive Cactus has been developed over several years, and has already been successfully used as an integral component of high-profile comparative genomics projects [16, 17, 18, 19, 20].

Progressive Cactus

The new Progressive Cactus pipeline is freely available and open source. The only inputs needed are a guide tree and a FASTA file for each genome assembly.

The key innovation of Progressive Cactus is to adapt the classic *progressive* strategy (used in collinear multiple alignment for decades) to a whole-genome alignment setting. Progressive aligners use a *guide tree* to recursively break a multiple alignment problem into many smaller sub-alignments, each of which is solved independently; the resulting sub-alignments are themselves aligned together according to the tree structure to create the final alignment. Progressive alignment has been successfully applied to whole-genome alignment before, for example by progressiveMauve [21] and TBA/MULTIZ [22]. Cactus now follows a similar strategy, with the key innovation being that Progressive Cactus implements a progressive-alignment strategy for whole-genome alignment using reconstructed ancestral assemblies as the method for combining sub-alignments. This strategy (analogous to MAVID's [23] strategy of using ancestral reconstruction in collinear multiple alignment) not only results in a much faster alignment runtime but also produces ancestral reconstructions.

Figure 1A shows the overall organization of the Progressive Cactus process. The guide tree, which need not be fully resolved (binary), is used to recursively split a large alignment problem (comparing every genome to every other genome) into many small subproblems, each of which compares only a small number (usually 2–5) of genomes against one another. The purpose of each subproblem is to reconstruct an ancestral assembly at each internal node in the guide tree, as well as to generate alignments between that internal node's children and its ancestral reconstruction. The ancestral assemblies are then used as input genomes in subproblems further up the tree, while the parent-child alignments are later combined to produce the full alignment. Two sets of genomes are considered: the children of the internal node (which we call the *ingroup genomes*), and a set of non-descendants of that node (the *outgroup genomes*). The ingroup genomes form the core alignment relationship being established at this node. The outgroup genomes serve to answer the question of what sequence from the ingroups is also present in the ancestor (whether an indel among the ingroups is likely a deletion rather than an insertion), and in how many copies (whether a duplication predates or postdates the speciation event the node represents). The outgroups also provide information for guiding the ancestral assembly by providing order-and-orientation information, as well as base-level information when generating ancestral sequences. These genome sets are used as the input to the main subproblem workflow (Figure 1B).

Each individual subproblem follows a procedure akin to the original Cactus process. The subproblem procedure begins with a set of pairwise local alignments generated via the

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

sensitive pairwise local-alignment program LASTZ [24]. These pairwise alignments are then filtered and combined into a cactus graph representing an initial multiple alignment using the CAF algorithm described in our earlier work [6] — though we note important changes to the filtering in the Methods and Extended Data Figure 1 — to attempt to recover the homologies that date to the most recent common ancestor of the ingroup genomes. The initial alignment is refined using the previously described BAR algorithm [6] to create a more complete alignment. The ancestral assembly is then created by ordering the blocks in this final alignment and establishing a most-likely base call for each column in each block. The resulting ancestral sequence is then fed into later subproblems (unless the root of the guide tree has been reached, which ends the alignment process).

As a practical matter, Progressive Cactus uses the Toil [25] workflow framework to organize and distribute its computational tasks. While genome alignment is a computationally intensive task, using Toil, we can break up the problem into small pieces that can work in heterogeneous compute environments, playing to the advantages of both cheap CPU-rich machines and more expensive memory-rich machines. Because it runs on Toil and supports container execution via Docker and Singularity [26], Progressive Cactus can be run on many different environments: single machines (for small alignments), conventional clusters, as well as commercial clouds.

Given the rate of arrival of new assembly versions and newly sequenced genomes, adding new information to an alignment without recomputing it from scratch is valuable, especially for large alignments where recomputing the entire alignment is often cost-prohibitive. Progressive Cactus, combined with special functionality in the HAL toolkit [27], therefore supports adding and removing genomes from the alignment by taking advantage of the tree structure of the progressive alignments it produces (see Methods, Extended Data Figure 2 and Extended Data Table 1).

Evaluation on simulated data

The Alignathon’s simulated datasets [14] have been aligned with many competing genome aligners and have a known truth set, providing a way to compare Progressive Cactus against other genome aligners. Progressive Cactus produces alignments with higher accuracy for both simulated primate (F1 0.989) and mammal (F1 0.795) clades than any aligner that participated in the Alignathon (Supplementary Tables 1 and 2), including the original version of Cactus.

To evaluate the improvements in quality and runtime of the alignments produced using the new progressive alignment strategy, we simulated the evolution of 20 30-megabase genomes using Evolver [28] along a tree of catarrhines. We ran two alignment strategies — one using a fully resolved binary guide tree (which takes full advantage of the new progressive mode) and one using a fully-unresolved star guide tree (which is similar to the originally published version of Cactus) — across variously-sized subsets of genomes roughly evenly spaced throughout the catarrhine tree. The alignments using the progressive strategy finished more quickly, with the speed improvement growing larger with the increasing number of species (e.g. a 15% reduction in runtime for 10 species and 48% for 20 species), due to its linear

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

runtime scaling, as opposed to the quadratic scaling of the star-tree (Figure 2A). The progressive strategy is also more accurate than the star strategy (Figure 2B) and maintains accuracy as the number of species (and therefore nodes in the tree) increases.

Effect of the guide tree

Since Progressive Cactus uses an input guide tree to decompose the alignment problem, the guide tree can potentially impact the resulting alignment. This could be problematic when the exact species tree relating the input set of genomes is unknown or controversial. However, we reduce any effect of the guide tree by including a great deal of outgroup information, including multiple outgroups when possible. To quantify the effect of the guide tree on a large alignment with an uncertain species tree, we created four alignments of a set of 48 avian species (Supplementary Table 3), which we subset down to a single chromosome (Chromosome 1). The avian species tree is still being actively debated [29, 30] and there are different plausible hypotheses, making birds an excellent test case with no single clearly correct guide tree. We aligned these birds using four different guide trees: two trees that represent two different hypotheses about the avian species tree (Jarvis et al. [30] and Prum et al. [29]), one consensus tree between the former two trees, and one tree that was randomly permuted from the Jarvis et al. tree (see Methods and Supplementary Figure 1). The four alignments were highly similar, with an average of 98.5% of aligned pairs exactly identical between any two different alignments (Extended Data Table 2).

We further examined whether these small differences in the guide tree impact some species more than others. For any pair of these 48 species, the F1 score for aligned pairs between the Jarvis et al. based and Prum et al. based alignments was at least 0.955 (Supplementary Figure 2). As an example, the phylogenetic relationship between the species *Cuculus canorus*, *Chlamydotis macqueenii*, and *Tauraco erythrolophus* is different in the Prum et al. based guide tree than the Jarvis et al. based guide tree (Supplementary Figure 3). The F1 score for aligned pairs within this clade between the two alignments was 0.972, lower but comparable to the score for a similar clade that had an identical phylogenetic relationship in both trees, 0.982 (for *Merops nubicus*, *Picoides pubescens*, and *Buceros rhinoceros*).

Effect of assembly quality on alignment

Our progressive approach means that the alignment between two genomes distant in the guide tree is informed by the reconstructions of the ancestral genomes along the path, which is in turn formed using data from other genomes in the tree. To evaluate the practical effect of differing quality of input assemblies, we created two alignments of 11 boreoeutherian species, 7 of which represented either high-quality assemblies in one alignment (using modern assemblers and often long-read data) or lower quality assemblies in the other alignment (usually using much older technologies) (Supplementary Table 4). The remaining 4 assemblies were held constant to facilitate a comparison between the two alignments. Despite alignment differences between the new and old assemblies (Supplementary Table 5), the alignment between these 4 assemblies was similar between the two alignments (e.g. 0.855 Jaccard similarity between induced pairwise human-dog alignments; Supplementary Figure 4), a level of similarity higher than seen between alignment strategies, indicating that

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

the progressive alignment strategy can tolerate poor assemblies. Reinforcing this, comparing the induced pairwise alignments of human-dog to direct pairwise alignments computed using the established Chains and Nets pipeline [31], we find the same level of Jaccard similarity for both the high and low-quality assembly alignments (Supplementary Figure 5). 82% of aligned pairs in the induced pairwise Progressive Cactus alignments were found in the Chains and Nets alignment and, vice versa, 78% of pairs in the Chains and Nets alignment were found in each Progressive Cactus alignment. Concordant results were found comparing human-mouse pairwise alignments (Supplementary Figure 5).

600-way amniote alignment

To demonstrate Progressive Cactus we present results from an alignment of 605 amniote genomes, relating in a reference-free manner a total of over 1 trillion bases of DNA across hundreds of millions of years of genome evolution (an estimated 35.4 neutral substitutions per site). The amniote-wide alignment combines two smaller alignments: one created for the initial release of the Zoonomia project [32], representing 242 placental mammals, and one for the Bird 10K project [33], which relates 363 avians. The overall topology is shown in Figure 3A. To our knowledge, this represents the largest whole-genome alignment yet created. Table 1 contains aggregate statistics on this alignment.

Coverage within the 600-way alignment closely tracks phylogenetic distance and genome size, e.g. a median coverage on human of 2.3 gigabases (Gb) from Euarchonta species, vs. 1.2 Gb from Laurasiatheria species and 1.0 Gb from Glires species (Figure 3B,C). The ancestral reconstructions within the 600-way alignment are highly complete, especially for functional sequence: 86% of human coding bases are represented in our reconstruction of the ancestor of all placental mammals, while 95% of chicken coding bases are represented in our reconstruction of the common ancestor of avians (Figure 3D,E). Due to the long branch length (~0.7 substitutions-per-site divergence between the two clades), the amniote (human-chicken) ancestral assembly has a much lower proportion of reconstructed sequence than its immediate children, the avian and eutherian ancestors, e.g. retaining 16.3% of chicken intron bases vs. 84.4% in the avian ancestor, and 7.2% of human intron bases vs. 56.5% in the eutherian ancestor. However, coding bases are still well-retained (86.8% from chicken and 58.7% from human). The ancestral assemblies consistently contain a relatively higher proportion of sequence for avians than for mammals even across similar phylogenetic distance, reflecting a much more conservative mode of genome evolution in avians as well as the lower repeat content and denser gene content of avian genomes [34].

The ancestral reconstructions provide a history of substitution, indel, and rearrangement events. Though this history is by its nature only a hypothetical reconstruction of the true history of genome evolution along the tree, it is accurate enough to be useful. To demonstrate the utility of the indel history, we examined rates of small (≤20 bp) insertion and deletion events in the 600-way alignment. As expected given previous studies [35, 16], the rate of small indels in any given branch was correlated with the rate of nucleotide substitution (an R^2 of 0.69 for insertions and 0.80 for deletions in avians, and 0.39 and 0.40 respectively for eutherians), though the relative rates remained much lower for insertions (1.2% of the substitution rate for both clades) and for deletions (2.4% and 1.2% of the

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

substitution rate for avians and eutherians respectively). Interestingly, we observe similar rates of deletions between eutherian and avian lineages, but evidence of a slightly increased rate of insertions in avians (Extended Data Figure 3A). The ancestral assemblies also represent even difficult-to-align regions such as transposable elements. We ran RepeatMasker [36] on several human ancestors, focusing on the recently-emerged L1PA6 family of L1 retrotransposons. When ascending the primate tree, approaching the origin of modern L1PA6 elements above the human-rhesus ancestor, L1PA6 elements appear increasingly similar to their consensus sequence (Extended Data Figure 3B; Supplementary Figure 6).

Despite its scale, sub-alignments of the 600-way are similar to smaller alignments of the same species. Within the 7.1 billion aligned pairs involving human, mouse, rat, or dog within the 600-way, 76.49% were present in an alignment less than a tenth the size (Supplementary Figure 7) — this similarity is in line with that observed between different alignments of these same species[14]. As expected, the alignments more strongly agree in functional regions, such as coding exons, than for the genome as a whole (Supplementary Figure 8). The size and fraction of functional elements reconstructed in ancestors shared between the 600-way and smaller alignments of mammals and, separately, avians were also highly similar (Supplementary Figure 9, Supplementary Figure 10).

To evaluate the relative accuracy of the progressive alignment process back to the amniote ancestor, human protein-coding transcripts and genes were mapped to the chicken genome using translated BLAT [37], translated BLAST [38], LASTZ [24], and the 600-way. Of 84,001 transcripts BLAT mapped 70%, BLAST mapped 80%, LASTZ mapped 67%, and Progressive Cactus mapped 74%. Both Progressive Cactus and LASTZ had much lower levels of multi-mapping (2–3% of transcripts) than either translated method (16%−51%) (Supplementary Tables 6, 7, 8). Comparing Cactus and LASTZ coding sequence mappings to the union of the translated alignments, both in terms of individual gene counts and coding and mRNA bases, Cactus has a marginally higher fraction of shared elements with the translated alignments than LASTZ (Supplementary Table 9). Supporting this result, comparing the median per-transcript and per-gene base-level Jaccard similarity of these mappings to chicken, Progressive Cactus, and LASTZ were most similar, and Progressive Cactus was more similar to translated BLAT and Blast than LASTZ (Supplementary Figures 11, 12; Supplementary Table 10). Both Progressive Cactus and LASTZ have higher levels of base-level similarity with existing Chicken annotations than either translated alignment method (Supplementary Table 11).

The Bird 10K species were also separately aligned with MULTIZ [22] using the chicken genome as the reference, allowing us to make a comparison between the two resulting alignments. Progressive Cactus aligned more total bases to chicken than MULTIZ (an average of 69.4% of the chicken genome compared to an average of 64.9%, for an average increase of 47 Mb). Since, unlike Progressive Cactus, MULTIZ is reference-biased, the difference is starker when looking at the number of bases aligned to a genome not used as the MULTIZ reference (an average of 79% of the zebra finch covered vs. 49.2%, for an average increase of 367Mb: see Figure 4).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Discussion

The Vertebrate Genome Project [39], Genome 10K [40], and the Earth BioGenome Project [41], among others, aim to release thousands of new, high-quality genome assemblies over the next decade. These projects will give us incredible insight into natural history but will need massive genome alignments. We have demonstrated that Progressive Cactus can create reference-free alignments of hundreds of vertebrate genomes efficiently. The Bird 10K [33] and Zoonomia [32] consortia are using this alignment for comparative analysis, for example, analyzing patterns of selection in unprecedented detail.

We focus on creating a reference-free alignment and ancestral reconstruction, allowing analysis of genome evolution throughout the entire tree rather than in comparison to one anointed reference. As the average assembly becomes ever more complete and accurate [39], the value of such a reference-free approach grows. Similarly driven by technology improvements, sequencing efforts will increasingly produce multiple, phased *de novo* assemblies from different individuals in a population [42]. Progressive Cactus has already proved useful for comparison between assemblies of the same species [20]. Alignments of such assemblies are essential for annotation [9] and variant characterization [43] and should prove useful for reference-free pangenome construction of the variation present in a population [44].

Data availability

The 600-way genome alignment is composed of data gathered for the Zoonomia mammalian genomes project and data from the Bird 10,000 genomes (B10K) project. All genomes have been archived in GenBank, spreadsheets containing all the accession numbers of the assemblies is provided in the supplementary material. The 600-way alignment is available in HAL format at https://alignment-output.s3.amazonaws.com/600way.hal. We also provide the subset of the alignment containing the Zoonomia genomes at https://alignment-output.s3.amazonaws.com/200m-v1.hal. The subset of the alignment containing the Bird 10K genomes is at https://alignment-output.s3.amazonaws.com/birds-final.hal. A visualization of the alignments and associated data is available by loading our assembly hub into the UCSC browser. By copying the hub link https://comparative-genomics-hubs.s3-us-west-2.amazonaws.com/600way_hub.txt into the “Track Hubs” page, the 605 genomes and associated tracks will be available. Note that the B10K consortium is organizing phylogenomic and other analyses with the avian alignment and sequencing data. We encourage persons to contact us for collaboration if they are interested in using this data for phylogenetic analyses.

Code availability

The Progressive Cactus pipeline is available at https://github.com/ComparativeGenomicsToolkit/cactus under the MIT license, version 1.0 is archived here: https://doi.org/10.5281/zenodo.3873410. The exact version of Progressive Cactus used for each of the analyses described above varies; for the commit used in each analysis see the Methods.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Methods

Evaluation on simulated data

20 primate genomes were simulated using Evolver, managed using the evolverSimControl (https://github.com/dentearl/evolverSimControl, commit b3236deb) pipeline. The root genome used was derived from 30 megabases selected from the hg19 genome, and is available at http://courtyard.gi.ucsc.edu/~jcarmstr/datastore/progressiveCactusEvolverSim.tar.gz along with the Evolver configuration files that were used. The species tree used for the simulation was obtained from a catarrhine subtree of the 100-way alignment tree available on the UCSC browser.

The tree used was, in Newick format:

```
(((((((Human:0.00655,Chimp:0.00684)anc0e:0.00122,Bonobo:0.00784)anc1e:0.003,Gorilla:0.008964)anc2e:0.009693,Orangutan:0.01894)anc3e:0.003471,Gibbon:0.02227)anc4e:0.01204,
(((((Rhesus:0.004991,Crab_eating_macaque:0.005991)anc5e:0.001,Sooty_mangabey:0.001)anc6e:0.005,Baboon:0.003042)anc7e:0.01061,
(Green_monkey:0.027,Drill:0.03)anc8e:0.002)anc9e:0.003,
((Proboscis_monkey:0.0007,Angolan_colobus:0.0008)anc10e:0.005,(Golden_snub-nosed_monkey:0.0007,Black_snub_nosed_monkey:0.0008)anc11e:0.004)anc12e:0.009)anc13e:0.02)anc14e:0.02183,
(((Marmoset:0.03,Squirrel_monkey:0.01035)anc15e:0.01065,White-faced_sapajou:0.009)anc16e:0.01,Nancy_Mas_night_monkey:0.01)anc17e:0.01)anc18e;
```

The alignments were generated using Progressive Cactus commit 51eb980b. The input files (the simulated genomes as well as input files and Progressive Cactus configuration file) are available at http://courtyard.gi.ucsc.edu/~jcarmstr/datastore/progressiveCactus.EvolverSim.CactusInput.EvenlySpread.tar.gz. A non-default configuration (included in the dataset) was used to change the alignment filtering in both runs to better support the high degree of polytomy in the star-tree runs. Four sets of 2, 6, 10, and 20 genomes were used, each of which were run three times to generate runtime estimates. The sets are as follows: 2 species: rhesus and marmoset; 6 species: rhesus, marmoset, gorilla, drill and black snub-nosed monkey, white-faced sapajou; 10 species: species from 6-species alignment and human, sooty mangabey, proboscis Monkey and Nancy Ma's night monkey; 20 species: all species.

The runtime statistics were gathered using the toil stats command (the overall Clock time was used, which represents CPU time spent across all jobs). To generate the recall and precision statistics, MAFs were exported for each run (using hal2maf from the hal package (https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d) with the --onlyOrthologs option using the rhesus genome as a reference) and compared to the Evolver MAF using mafComparator (https://github.com/dentearl/mafTools, commit 82077ac3).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Comparison using Alignathon data

For comparison against other genome alignment methods, we aligned data (both the simulated “primates” and “mammals” datasets) used in the Alignathon using Progressive Cactus. For comparison, we downloaded all the original Alignathon entries in MAF format. We used the original Alignathon analysis workflow (https://github.com/dentearl/mwgAlignAnalysis,commitdf98753) to reanalyze the MAFs, with the output of the newest Progressive Cactus version added, to generate the precision/recall statistics (which we extracted from the comparison against the most recent common ancestor (MRCA) truth set). The simulated-mammal results are shown in Supplementary Table 1, and the simulated-primates results are shown in Supplementary Table 2.

Evaluation of the effect of the guide tree

The guide-tree analysis was performed on a set of 48 bird genomes originally published in 2014 by Jarvis et al. To reduce the amount of alignment work required, we subset these genomes down to the size of only a single chromosome, chicken chromosome 1 (by removing any contig or scaffold which had less than 20% of its sequence alignable to chicken chromosome 1). We used Progressive Cactus commit 36304707 for all alignments in this analysis.

The Prum and Jarvis topologies were adapted from Prum et al. and Jarvis et al., respectively. The “permuted” topology was generated starting from the Jarvis topology, via 3 randomly chosen subtree-prune-regraft operations followed by 3 random nearest-neighbor-interchange operations. Each of these three topologies had branch-length estimates performed using phyloFit from the PHAST package (https://github.com/CshlSiepelLab/phast, commit 52e8de9) based on fourfold-degenerate sites of BUSCO orthologs. Finally, the “Consensus” tree was produced as a strict consensus of the Jarvis and Prum trees (collapsing all groupings that were not the same in both trees) using the ape::consensus method from the APE R package [47]. The branch-lengths for this tree were generated from the fitted branch lengths for the two input trees, using the consensus.edges function of the phytools R package [48]. The four final trees that were used in the four Progressive Cactus alignments are shown in Supplementary Figure 1, and available in supplementary data in Newick format.

We further focused on the alignments with guide trees based on Jarvis / Prum (Supplementary Figure 3) to establish what alignment differences resulted from different phylogenetic hypotheses. Supplementary Figure 2 shows a refinement of the overall alignment-to-alignment F1 scores shown in Extended Data Table 2, showing the F1 scores for each species pair between the Jarvis- and Prum-based alignments. Each pair of species has an F1 score between Jarvis- and Prum-based alignments of at least 0.955.

Impact of assembly quality on alignment quality

We aligned two sets of 11 boreoeutherian genomes: one where 7 of the species were represented by relatively low-quality assemblies, and another where the same 7 species were represented by higher-quality assemblies; the assemblies used are listed in Supplementary Table 4. The remaining four genomes had the same assemblies in both alignments to

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

facilitate comparison (human, hg38; mouse, mm10; rat, rn6; and dog, canFam3). We used Progressive Cactus commit 36304707 for all alignments in this analysis.

Generation of the 600-way alignment

The Zoonomia alignment was composed of two sets of mammalian genomes: newly assembled DISCOVAR assemblies and Genbank assemblies. The DISCOVAR genomes were masked with RepeatMasker commit 2d947604, using Repbase [49] version 20170127 as the repeat library and CrossMatch as the alignment engine. The pipeline used is available at https://github.com/joelarmstrong/repeatMaskerPipeline,commita6ad966. The guide-tree topology was taken from the TimeTree database [50] (using release current in October 2018), and the branch lengths were estimated using the least-squares-fit mode of PHYLIP (http://evolution.genetics.washington.edu/phylip/getme-new1.html, version 3.695) [51]. The distance matrix used was largely based on distances from the 4d site trees from the UCSC browser [52]. To add those species not present in the UCSC tree, approximate distances estimated by Mash (https://github.com/marbl/Mash, commit 541971b) [52] to the closest UCSC species were added to the distance between the two closest UCSC species. We used the hal package to process the hal file (https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d).

The final guide tree is embedded in the HAL file, and available using the halStats --tree command.

The 363 assemblies in the B10K alignment comprised four sets: 236 newly sequenced species for the “family” phase of the project, assembled using SOAPdenovo2 and AllpathsLG, 42 assemblies already sequenced from the “order” phase of the project, 36 assemblies taken from GenBank, and 49 assemblies contributed by other research groups. For the avian guide-tree, we used a tree that the B10K consortium derived as preliminary data from ultraconserved elements.

Both alignments were run on the AWS cloud over the course of 3 weeks for the avians and 2 months for the mammals, using a maximum of 240 c3.8xlarge instances and 20 r3.8xlarge instances. Because Toil’s autoscaling mode was used, this capacity was only fully utilized during the initial phase of the alignment, when the potential for parallelism was at its highest.

The 600-way alignment was formed by aligning the two roots of the B10K and Zoonomia alignments, using the xenTro9 (frog), latCha1 (coelacanth), and danRer11 (zebrafish) assemblies as outgroups. This created a “linker” alignment connecting the roots of the two alignments. The B10K and Zoonomia alignments were then added to this linker alignment using the halAppendSubtree command.

Micro-indel events within the 600-way

We extracted all insertion and deletion events by running the halBranchMutations (https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d) tool on every branch in the 600-way alignment. The ungapped insertion and deletion calls (represented by “I” and “D” respectively within the output file) were filtered so that only calls spanning less than 20bp

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

(in the child for insertions, and the parent for deletions) were counted. The rate for each branch was then obtained by dividing the count of these micro-indel events by the total amount of sequence present in the child.

Repetitive elements within ancestral sequences

We ran RepeatMasker (https://github.com/rmhubley/RepeatMasker, commit 2d947604) on all ancestral assemblies of human within the 600-way alignment (using RepBase [49] version 20170127, selecting the “primate” repeat library and choosing CrossMatch as the alignment engine). We additionally ran the same pipeline against human (as existing annotations used the “Homo_sapiens” repeat library). All ancestors up to human-rhesus had over 78% of the human complement of L1PA6 elements (Supplementary Figure 10).

Human/chicken transcript alignment protocols

Protein-coding transcript annotations were obtained from the UCSC Genome browser [51] tables. Human annotations are GENCODE V34 on hg38 (GRCh38/GCA_000001405.27) and chicken annotations are Ensembl 85 on galGal4 (GCA_000002315.2). Predicted RNA sequences for each protein-coding transcript are extracted from the genome. Only gene annotations on the primary assemblies were used, those on alternate loci, patches, and assembled sequences were dropped. This results in 84,001 transcripts in 19,695 genes for human and 15,328 transcripts in 14,499 genes for chicken. The human transcripts were then mapped from the human genome to the chicken genome. The steps for each method are outlined below, although the actual execution was done by partitioning the data and using a cluster. Command-line tools from the UCSC Genome Browser group and programs used came from: https://github.com/ucscGenomeBrowser/kent, commit 8a8d921, https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d, ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/2.10.0/, version tblastn: 2.10.0, and https://github.com/lastz/lastz, version 1.03.54.

BLATX transcript alignment protocol

The *BLATX* alignments were created using protein-translated mode to align the mRNAs to the target genome with *BLAT* version 36×5. They were then filtered following the same protocol the UCSC Genome Browser uses for creating the other species RefSeq alignments:

```
blat -noHead -q=rnax -t=dnax -mask=lower <dest-genome.2bit> \
<src-rna.fa> <dest-rna-raw.psl>
```

We then filter to get near-best in genome. Alignment to chicken uses near best filter of -localNearBest=0.010 while to human it uses -globalNearBest=0.010:

```
faPolyASizes <src-rna.fa> <src-rna.polya>
pslCDnaFilter <nearBestOption> -minId=0.35 -minCover=0.15 -minQSize=20 \
-ignoreIntrons -repsAsMatch -ignoreNs -bestOverlap \
-polyASizes=<src-rna.polya> <dest-rna-raw.psl> <dest-rna-mapped.psl>
```

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

The transMapPslToGenePred command is then used to project the original CDS onto the alignment.

TBLASTX transcript alignment protocol

The *TBLASTX* alignments were created using the protein-translated *tblastx* program to align the mRNAs to the target genome with *BLAST+* version 2.10.0+.

The database is created using the repeat masking from the UCSC Genome Browser genomes to match what is used within the *BLATX* methodology above:

```
convert2blastmask -in <dest-genome.fa> -masking_algorithm repeat \
-masking_options “repeatmasker, default” -outfmt maskinfo_asn1_bin \
-out <dest-genome.mask>
makeblastdb -dbtype nucl -in <dest-genome.fa> -mask_data <dest-genome.mask>
```

The mRNAs are aligned and the resulting XML converted to PSL format, filtering to an e-value threshold of 0.01. These are then chained using a program the UCSC group developed for chaining *BLAST* alignments:

```
tblastx -db <dest-genome.fa> -db_soft_mask 40 -outfmt 5 -query <src-rna.fa> \
-out <dest-rna-raw.xml>
blastXmlToPsl -eVal=0.01 <dest-rna-raw.xml> <dest-rna-raw.psl>
simpleChain -outPsl -maxGap=75000 <dest-rna-raw.psl> <dest-rna-chained.psl>
```

The alignments produced are then filtered in the same manner as the *BLATX* alignments.

LASTZ transcript alignment protocol

Both the *LASTZ* and *Cactus* transcript mappings use the *TransMap* [54] projection alignment algorithm to project transcript annotation between genomes.

The *LASTZ* alignment chains and nets [55, 56] were obtained from the UCSC Genome Browser downloads. These were then filtered to produce a set of syntenic mapping chains using these steps:

```
netFilter -syn <genomes.net> <syntenic.net>
netChainSubset -wholeChains <syntenic.net> <genome.chain> <mapping.chain>
```

Cactus transcript alignment protocol

The *Cactus* alignments are extracted for all primary chromosomes from the HAL file and chained using the same chaining algorithm as the *LASTZ* chains, with the –noDupes option having a similar effect as the syntenic net filtering:

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

```
halLiftover --outPSL --noDupes 600way.hal <srcOrganism> \
<srcChroms.bed> <destOrganism> <src-dest.psl> <genome.psl>
axtChain -psl -linearGap=loose -scoreScheme=HoxD55.q <genome.psl>
<mapping.chain>
```

The *TransMap* protocol is used for both the *LASTZ* and *Cactus* mapping chains to produce alignments of the transcripts to the other genomes. This used the *pslMap* command to do the mapping and *pslRecalcMatch* to update the statistic in the alignments:

```
pslMap -chainMapFile <src-rna.psl> <mapping.chains> <dest-rna-over.psl>
pslRecalcMatch <dest-rna-over.psl> <dest-genome.2bit> <src-rna.fa>   <dest-
rna-raw.psl>
```

The alignments produced are then filtered in the same manner as the *LASTZ* alignments.

Transcript and gene alignment subsets and comparison

To facilitate the comparative analysis of the alignment methods, we created reduced sets of the alignments using two different approaches.

While both *BLATX* and *TBLASTX* will align UTR, the strength of protein-translated methods is in recognizing distant coding sequence relationships. We have previously shown [53] alignment projection-mapping methods align more UTR bases than translated methods. To facilitate comparisons, CDS alignments from each method were created by trimming the RNA alignments to contain only the CDS regions as defined by the human annotation set.

While mapping all transcripts is useful, particularly for understanding the utility of the methods in assisting genome annotation, individual transcripts overlap, biasing assessment of transcribed mappings to genes with larger transcript numbers. To remove most of this base multiplicity from comparisons, in addition to showing full transcript results, subsets of the alignments are created using only one representative transcript per gene. For the full RNA alignments, the longest RNA for each gene was chosen, with the CDS alignments choosing the transcript with the longest CDS. The biology of overlapping gene structures and the ambiguities in defining genes cause around 4% of genomic bases to appear in more than one gene in the RNA and 3% in the CDS gene sets due to overlap.

Individual pairwise alignments were compared at the base-level, consistent with the earlier comparisons reported. Briefly, alignment similarity is computed by comparing the set of shared aligned pairs. That is, a pairwise alignment can be viewed as a set of aligned base pairs, each a coordinate from the source (human) and target (chicken) genome. The Jaccard index is, in this context, the number of aligned pairs identical between the two alignments divided by the union of all aligned pairs in the two alignments. It is worth noting that translated alignments are encoded for comparison using their induced base-level alignments. Transcripts or genes that aren’t aligned by either of the aligners being compared are assigned Jaccard indices of zero.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

To account for human bases which map to multiple bases in chicken (which occurs frequently for the translated alignment methods that include very distant, fragmented, paralogous alignments, but much less often for the non-translated methods), when comparing the alignments of an mRNA or CDS between two methods, if either or both methods produces multiple alignments, we pick the pair of mappings (one from each method) with highest shared similarity to report. This generally has the effect of removing distant paralogies from the comparison.

Progressive Cactus Methods

Progressive Cactus builds upon the original Cactus program, in particular the CAF and BAR algorithms, which are described in detail in the original publication. In overview, the CAF algorithm (short for Cactus Alignment Filter) is an algorithm designed to construct a sequence graph from an input set of local alignments (in the Progressive Cactus pipeline computed using LASTZ). We omit a complete definition here, but a sequence graph represents the alignment of a set of nucleotide strings. It can formally be represented using a bidirected or biedged graph [57, 58, 59] (Supplementary Figure 13(A)). Larger nucleotide strings are encoded as walks through sequence graphs (Supplementary Figure 13(B)); in the biedged representation an alignment between two or more substrings is represented by both strings visiting a common sequence edge; in Progressive Cactus each sequence edge represents an alignment ‘block’, a set of oriented substrings in the set of input strings which are considered to be gaplessly aligned. A key property of alignments represented by sequence graphs is that the alignments they represent are equivalence relations: that is, alignments are transitive, reflexive and symmetric. The core challenge the CAF algorithm addresses is sub-selecting which local alignments from the input set to include in the sequence graph, because typically a collection of local alignments computed with a tool like LASTZ will contain numerous transitive inconsistencies which when combined will create implausible, high degree alignment blocks in the sequence graph. The CAF algorithm uses the 3-edge connected components of a sequence graph to define a restricted form of cactus graph such that there exists a homomorphism from the alignment blocks in the sequence graph onto the resulting cactus graph (Supplementary Figure 13(C)). In the constructed cactus graph each edge is a member of exactly one simple cycle. These simple cycles correspond to ‘chains’ of alignment blocks, maximal sequences of blocks whose aligned substrings appear in the same order and orientation in the input strings. The CAF algorithm iteratively filters the input set of alignments to remove local alignments that create short simple cycles in the cactus graph, this is achieved by deleting alignment blocks from the sequence graph involved with these short cycles. The result of the CAF algorithm is a filtered set of local alignments represented using a sequence graph. To add to the output sequence graph of the CAF algorithm the BAR algorithm constructs a detailed alignment by extending gapped alignments from the ends of each alignment block, using a greedy approach to force consistency between the alignments constructed starting from connected alignment blocks. In Progressive Cactus the CAF and BAR algorithms are applied to create an alignment of the corresponding set of in-group and out-group species for each internal node of a guide tree.

Below we provide updates on the changes made to Cactus to create Progressive Cactus.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Preliminary repeat-masking

Progressive Cactus requires input genomes to be soft-masked, but often repetitive sequence goes unmasked due to poor masking or incomplete repeat libraries for newly-sequenced species. This can negatively affect alignment runtimes (as alignments need to be enumerated to and from all copies of a repetitive sequence) and impact quality. For this reason, we mask overabundant sequence before alignment, using a strategy not based on alignment to repeat consensus libraries, but on over-representation of alignments. We first divide each genome into small, mutually overlapping chunks. For each chunk, we align it to itself and a configurable amount of other randomly sampled chunks (currently 20% of the total pool). To avoid combinatorial explosion due to unmasked repetitive sequence, we use a special mode of LASTZ which stops exploring alignments from any region early if a maximum depth is reached (using the flag --queryhsplimit=keep,nowarn:1500 which stops after a high-scoring-pair depth of 1500). We then soft-mask any region covered by more than a configurable number of these alignments (currently set to 50). Further details can be found in the src/cactus/preprocessor section of the Progressive Cactus codebase. Though the preprocessing step is automatically run as part of the pipeline, we also provide a cactus_preprocessor utility to run only the preprocessor without producing a full genome alignment.

Local alignment and outgroup selection

The alignment process for each subproblem begins with a series of local alignments generated using LASTZ. The local alignments fall into two sets: a set of all-against-all alignments among the ingroup genomes, and a set of alignments from ingroup genomes to outgroup genomes. We have found outgroup selection to be absolutely crucial in creating an acceptable ancestral reconstruction: any missing data or misassembly in the outgroup that causes a true deletion in one of the ingroups to be misinterpreted as an insertion in others will mean that the ancestor contains less sequence than it ought to. This missing sequence in turn impacts the alignment between the entire subtree below the reconstructed ancestor and the entire supertree above it: the missing sequence will never be aligned between the subtree and supertree. To avoid this we attempt to use multiple outgroup genomes in each subproblem (by default, the 3 nearest outgroup genomes, as measured by branch-length). Naively aligning each ingroup against multiple outgroups would significantly increase the computation time; to avoid this we note that in general any region already containing an outgroup alignment benefits very little from aligning an additional outgroup. Therefore, we iteratively align each ingroup against one outgroup at a time, pruning away any ingroup sequence already covered by the previous outgroup alignments. In this way the computational cost is reduced to be far less than naively aligning against the entire outgroup set, while still retaining nearly all of the benefit. In addition, we allow the user to designate certain genomes in the input as being of particularly high quality; these are chosen as outgroups if possible to avoid problems with missing data in regions like mitochondrial or sex chromosomes that are often missing from some assemblies but not others.

Paralogy resolution

Users of a genome alignment are often interested in *orthology* relationships, rather than all *homology* relationships, between a set of sequences. For example, when comparing human

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

and chimpanzee KZNF genes, providing an alignment from each gene to the over-400 [59] homologous KZNF genes in the other genome is nigh-useless; the user is likely interested in only the orthologous copy or copies (in the case of a lineage-specific duplication) in the other genome. For this reason, Progressive Cactus alignments are capable of representing complex orthology/paralogy relationships, with an ability to display the alignment(s) labeled as orthologous, but also the option for a user to request alignments to paralogs at a customizable divergence-time threshold. This is achieved by implicitly producing a gene tree as the alignment is built, albeit with some restrictions, namely that a duplication event is represented by multiple regions in the child(ren) aligned to a single region in the parent species. This forbids the representation of gene-tree-species-tree discordance as would occur in incomplete lineage-sorting or horizontal transfer, as well as the exact ordering of multiple duplication events along a single branch. The restricted problem we solve at each subproblem step is that each alignment block should represent all regions orthologous to a single region of the ancestral sequence, possibly multiple per species; we make no attempt to fully resolve the gene tree when multiple duplications take place along a single branch. However, this still requires resolving the timing of all duplication events to the lineages of the tree: duplicated sequences whose coalescence precedes the speciation event represented in the subproblem should be split, while those following the speciation event should be kept together.

To achieve the desired alignment blocks in each subproblem, in constructing the initial sequence graph during the CAF algorithm Progressive Cactus greedily chooses which pairwise alignments to include in an effort to prevent paralogous alignments between the ingroup species. We developed two algorithms. Both are greedy algorithms designed to rank the pairwise local alignments and then iteratively add the alignments to the graph, at each step choosing to accept or reject the addition of alignments to the graph. Each added alignment ‘glues’ together two alignment blocks, splitting existing alignment blocks as necessary and merging the resulting two alignment blocks into one new block in the graph (Supplementary Figure 14).

The first algorithm, which was used in previous, beta versions of Progressive Cactus, relied on an outgroup-based heuristic to resolve duplication timing. This heuristic, which we term “single-copy outgroup filtering”, first sorts all the LASTZ alignments by their score in descending order. Then, starting from the highest-scoring alignment, it iteratively adds one alignment at a time to the sequence graph, rejecting the gluing of any two blocks if the resulting alignment block would contain two or more substrings from the same outgroup genome. In this way the heuristic refuses to glue blocks when the resulting block would contain homologies that imply duplications in the outgroups. These self-homologies within the outgroup would necessarily involve duplication events that occurred above or outside of the subtree rooted at the MRCA of the ingroup genomes. Since the goal at each progressive step is to determine (the transitive closure of) orthology relationships within this subtree, refusing these outgroup self-homologies proves useful for assigning orthology between ingroups. Unfortunately, this method is very sensitive to incomplete outgroup assemblies (containing an incorrect number of copies of a duplicated region) or variation in the similarity between closely related paralogs, causing assignment to the wrong copy. As seen

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

in Extended Data Figure 1, this filtering method tended to resolve duplications far too early, often causing paralogs to be called orthologs.

To remedy this problem, we developed an improved duplication-timing method, which we term “best-hit filtering”. The method preprocesses the local alignments to define for every base in every input genome a ranking by score of the local alignments that overlap it. The sequence graph is then built by first including the highest-scoring alignment for each base in each genome. We refer to this highest-scoring set as the set of ‘primary’ alignments and the remaining alignments the ‘secondary’ alignments. Note this definition is asymmetric: a pairwise alignment may be primary for one of the substrings it aligns and secondary for the other. All primary alignments are added to the initial graph unconditionally because they represent the most likely ortholog relationship (or in the case of multiple orthology, likely a random ortholog) (Supplementary Figure 15). The set of primary alignments represents a conservative set of alignment relationships that should include nearly no alignments to ancient paralogs. However, in regions that have undergone many rounds of lineage-specific duplications (which should all be aligned together in the restricted duplication-timing problem we described above), the set of primary alignments will often by chance not align all copies together. For this reason, after adding the primary alignments we iteratively add secondary alignments, going in descending order of score, rejecting any secondary alignment that would glue together any two existing blocks that both contain sequences from the same outgroup species (similarly to the “single-copy outgroup filtering” method) — this allows lineage-specific duplications of ingroup genomes to correctly land in the same block, while avoiding merging blocks from likely-paralogous alignments.

Of the two methods, the newer best-hit filtering removes many more likely-paralogous alignments, especially to closely-related genomes, while leaving approximately the same amount of sequence covered by at least one alignment. For example, we ran two versions of Progressive Cactus, one using the best-hit filtering and one using the outgroup filtering (commits 450da74 [best-hit filtering] and aca859f [outgroup filtering]), using the following tree:

```
(((((Human:0.006969,Chimp:0.009727):0.025291,Rhesus:0.044568):0.07,Tree_shrew
:0.19):0.03,(Kangaroo_rat:0.17,
(Mouse:0.072818,Rat:0.081244):0.11):0.150342):0.02326,
((Dog:0.07,Cat:0.07):0.087381,
((Pig:0.06,Cow:0.06):0.104728,Horse:0.05):0.05):0.04);
```

Comparing the best-hit filtering alignment and the one using the single copy outgroup filtering, the amount of human sequence mapping to two or more places in the chimpanzee genome was reduced from 6.1% to 2.6%, while the total amount of human covered by chimpanzee actually increased due to the removal of ancient homologs, simplifying the initial alignment relationships (see Extended Data Figure 1A,B for an example visualization and Extended Data Figure 1C for aggregate statistics).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

The alignment files are accessible in the URLs listed at Supplementary Table 12, and the assemblies used are listed in Supplementary Table 13. Coverage statistics from the resulting alignments were obtained using the halCoverage tool (https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d). To confirm that these improvements were likely caused by the removal of paralogous rather than orthologous alignments, we compared phylogenetic trees implicit in the columns of HAL alignments to independently re-estimated approximately-ML trees produced by FastTree (http://www.microbesonline.org/fasttree/, version 2.1.11) [61] for the same regions. The duplication-timing evaluation was performed using a custom pipeline (https://github.com/joelarmstrong/treeBuildingEvaluation) designed to sample columns from a HAL file and evaluate their trees against an independently re-estimated tree of the same region. For this analysis we used the two 12-boreoeutherian alignments described above, sampling 10,000 columns from the human genome. The comparison trees were built from a context of 1000 bases around the entries in each sampled column using FastTree 2.1.10 and the -gtr -nt options. Only duplicated columns were counted in the final output (columns containing no duplications did not count in the results). The coalescence pairs were evaluated using the --onlySelf option, meaning that only pairs that included the sampled site were counted in the results. To avoid weighting columns with a high number of copies per genome more than columns with a low number of copies per genome, only a single coalescence was randomly sampled per column.

Since HAL does not produce a fully binarized history of duplication events, we compared the most recent common ancestor (MRCA) of randomly selected pairs of sites from genomes containing a duplication within the column. If the MRCA species in the HAL tree is a descendant of the MRCA species within the reconciled ML tree, that implies that there are paralogs represented as orthologs within the HAL tree (since a duplication event must have been resolved too early). Similarly, if the MRCA species within the HAL tree is an ancestor of that within the reconciled ML tree, a duplication event must have been resolved too late in the HAL tree, implying additional false loss/deletion events. The number of paralogous alignments (represented by the coalescence time between duplicated sequences being too “early” in the HAL tree relative to the ML tree) in the alignment of the 12 boreoeutherian genomes was clearly reduced (46% in the outgroup filtering vs 26% in the best-hit filtering) (Extended Data Figure 1D).

We separately ran the Comparative Annotation Toolkit (CAT; https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d) [9] on identical human, chimpanzee, and gorilla assemblies (hg38, panTro6, and gorGor5 assemblies) in two alignments using the outgroup and best-hit filtering methods. We ran using the GENCODE V30 gene set [62]. We projected the transcripts solely via transMap without the use of the AUGUSTUS modes. Multiple-mapping statistics as well as the gene composition of the final gene set were taken from the filter_tm_metrics.json file in the CAT output.

Not only was CAT less likely to identify a human gene in multiple chimp loci using the best-hit filtering (e.g. 6.5% vs. 9.8% multiple-mappings across all genes in chimp, and 5.9% vs. 13.8% for the recently-duplicated KRAB zinc-finger gene family) (Extended Data Figure 1E), but as a result orthologs for 104 more human genes were identified in the output gene set for chimp (182 in gorilla) (Supplementary Table 14). This is likely because tens of

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

thousands of fewer paralogous transcripts were filtered out in the initial filtering phase of CAT (Supplementary Table 15), reducing confusion about which transcript projection to put into the gene set.

Removing recoverable chains

The original CAF algorithm was focused on removing small rearrangements while retaining as much of the original alignment relationships as possible in the filtered cactus graph. However, because the input local alignments are insensitive, the original alignment relationships are likely to have missed certain homologies. This can result in what we term *incomplete blocks*: blocks that contain some alignment relationships but are missing others, i.e. are proper subsets of the corresponding "true" alignment block. In our anchor-and-extend process, once a block becomes an anchor it can never be modified. As a result, these incomplete blocks will remain incomplete: they prevent the true alignment relationship from being found, even if an adjacent syntenic anchor block is complete and contains all desired alignment relationships. These problematic incomplete blocks become more prevalent at longer evolutionary distances: the local aligner will miss more true homologies at increasing distances, causing more incomplete blocks and in turn a far worse alignment.

To remove these incomplete blocks, Progressive Cactus originally relied on a heuristic that identified blocks that were "likely" to be incomplete, removing blocks that did not have alignment relationships between all ingroups. However, this heuristic performed poorly in the presence of deletions or missing data: any large deletion in one ingroup could cause huge stretches of the other ingroup(s) to be left unaligned. To remedy this, we have developed a new alteration to the CAF algorithm, one that now focuses on maximizing the potential size of the alignment graph *after* extension as opposed to *before* extension. We call this addition *removing recoverable chains* because it identifies chains in the cactus graph that represent alignments which could be recovered by the BAR algorithm extension process.

The algorithm is applied as a post-processing step after the CAF process, which proceeds as normal. After the cactus graph is created and filtered, the algorithm identifies *recoverable blocks*. Each block is composed of segments, each of which represents a non-overlapping region of a sequence and which strand is being aligned; we briefly review the necessary terminology, but see [63] for additional context. We call a segment *a left-adjacent* to another segment *b* if *a* represents the positive strand and *b* comes before *a* in their sequence and there is no other segment between them. Similarly, we call *a* left-adjacent to *b* if *a* is on the negative strand and *a* comes before *b* in their sequence ordering with no other intervening segment. If *a* is left-adjacent to *b*, then *b* is *right-adjacent* to *a*.

A block is called *recoverable* if, in the case that the block was removed, all its regions would be contained entirely within a single end alignment in the BAR extension phase. The end alignments are identified by looking at all unaligned sequences between the adjacent segments of a single *end* of a block: in short, two end alignments are created for every block, one for all sequences between each segment and its left-adjacent segment, and similarly for the right-adjacent segments. In practice, this means that for some block A, it is recoverable if all its segments are all left- or right-adjacent to segments from the same block $B \neq A$.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Whether a block is recoverable depends only on its immediate neighboring blocks. However, it is interesting to consider the maximum set of recoverable blocks, and, by contrast, of unrecoverable blocks — these unrecoverable blocks represent a minimal set of anchors that can be extended from to recover the alignment relationships from the original sequence graph as well as potential additional alignment relationships.

Since the chains and nets within the cactus graph represent a hierarchy of the rearrangements implicit in the alignment, they are helpful for finding a smaller set of anchors to extend from. We consider what anchors could provide recoverability to a block: if a block A's segments would lie within the end alignment of B if all the recoverable blocks between B and A, including A, were destroyed, we call A *recoverable given* B. The relationship is transitive: if block A is recoverable given block B, and B is recoverable given C, then A is recoverable given C. All blocks in a chain are recoverable given each other, since all blocks in a chain are collinear with each other, potentially with intervening rearrangements located further down the chain/net hierarchy. Similarly, if any block in a chain is recoverable given another block above the chain in the chain/net hierarchy, the entire chain is recoverable given that block. Due to this fact, in order to determine the recoverability status of all blocks, we only have to examine the blocks at the ends of chains and their immediate neighbors, rather than every block.

Though in principle we would need to keep only one block within even unrecoverable chains (since all other blocks within the chain would be recoverable given that single block), to save computational effort in realignment we only destroy or keep entire chains as a unit. In the same spirit, to avoid spending needless effort when the chain is recoverable but very likely is not *incomplete*, we apply a heuristic and do not remove chains that contain the same number of copies in all ingroups and outgroups.

After identifying and removing all recoverable blocks, some blocks previously marked unrecoverable may become recoverable (because adjacent blocks were removed). For this reason, we run the process of identifying and removing recoverable chains multiple times in a loop, until either no recoverable chains are identified or a limit on the number of cycles is reached. The structure of the cactus graph may change after removing recoverable blocks, so we recompute the cactus graph after every removal step. The process that is followed is described in pseudocode as follows:

```
function RemoveRecoverableChains(G, n)
for 1 … n do
cactusGraph ← CreateCactusGraph(G)
RecoverableChains ← ∅
for chain C in cactusGraph do
if
▷ A single adjacent end offers the potential for recoverability
(|C.leftAdjacencies| = 1 or |C.rightAdjacencies| = 1)
▷ Shared adjacencies indicate a non-recoverable rearrangement
and C.leftAdjacencies ∩ C.rightAdjacencies = ∅
```

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

```
▷ Links between chain ends indicate a non-recoverable duplication
and C.leftEnd ∉ C.rightAdjacencies then
RecoverableChains ← RecoverableChains U {C}
end if
end for
if |RecoverableChains| = 0 then
break
else
Destroy each chain in RecoverableChains
end if
end for
end function
```

Improvements from removing recoverable sequence

To quantify the effect that the process of removing recoverable chains (described above) had on real alignments, we ran alignments on a set of 9 Euarchontoglires genomes with the feature turned on and off. The tree used was:

```
(((((((human:0.00877,gorilla:0.008964):0.009693,orang:0.01894):0.015511,rhesus:
0.037601):0.07392,tarsier:0.1114):0.034014,tree_shrew:0.19114):0.002,
(kangaroo_rat:0.171759,
(chinese_hamster:0.14,mouse:0.132282):0.11015):0.114051)euarchontoglires:0.020593,(cow:0.18908,dog:0.13303):0.032898);
```

We used Progressive Cactus commit 56874bde, with the --root euarchontoglires option so that cow and dog were used only as outgroups. Coverage on human increased for all genomes when recoverable chains were removed, especially for those most distant from human (Supplementary Figure 16). This likely reflects the fact that though the losses caused by not removing recoverable chains in any single subproblem are relatively small, they can compound to be quite significant in large alignments since many subproblems are involved in creating the alignment between distant species (such as human and mouse, which are separated by 7 internal nodes in this tree).

Ancestral genome reconstruction

The core of what makes the progressive alignment algorithm possible is the ancestral reconstruction generated in each subproblem. This assembly serves as a summary of each subproblem alignment; the alignable sequence between the genomes in the subtree below the ancestor, as well as that alignable between the subtree and the supertree above the ancestor, is all present in the ancestral reconstruction. The ancestral sequence contains a base for each column in all blocks which contain an alignment between two ingroups and/or an ingroup and an outgroup — any alignment purely between outgroups is discarded. The order and orientation of the blocks relative to one another is chosen via a previously published algorithm for ordering a pangenome [64].

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

The identity of the ancestral bases is inferred via maximum-likelihood on a Jukes-Cantor model [65] of evolution using Felsenstein's pruning algorithm [66] on the subtree of the guide tree induced by the genomes in the subproblem. These base-calls are generated as the alignment is being made, so they necessarily take only a part of the alignment information into account and may be different than the ideal base-calls would be if taking into account information across the entire alignment. However, we provide a tool, ancestorsML, distributed as part of the HAL toolkit, that re-estimates ancestral base-calls after completion of the alignment if desired.

Adding a new genome to an existing alignment

Progressive Cactus supports adding a new genome to an existing alignment by taking advantage of the tree structure of the progressive alignments it produces. There are three ways that a new genome can be added to an alignment, depending on its phylogenetic position relative to the existing genomes: 1) as outgroup to all the existing genomes in the alignment, 2) by being added as a new child of an existing ancestral genome in the alignment, or 3) by splitting a branch in the existing alignment, creating a new internal node and two new branches (Extended Data Figure 2). Progressive Cactus allows adding a new genome in any of these ways, though the details differ (as described below). Assemblies can be replaced with new versions by simply deleting them and adding the new assembly in as a leaf. Adding multiple genomes is possible, either iteratively or (if the new genomes are monophyletic) by aligning together the new genomes and adding in the ancestral clade root.

Adding a genome as an outgroup is straightforward since it follows the normal progressive process: the root of the existing alignment and the new genome can be aligned together into a supertree alignment in which the existing subtree alignment can be appended to. A genome can be added as a new child of an existing internal node by simply aligning the new child, its siblings, and its parent together, without inferring a new ancestral genome. Adding a genome by splitting an existing branch is the least straightforward, but is key if the topology of the alignment or the accuracy of the ancestral genomes is important. To add a genome to an existing alignment, two subproblems are required: one relating the new genome and its new sibling in the target tree, constructing the ancestral genome that will split the existing branch, and one relating this new ancestral genome, its sibling, and its parent.

After the addition of a new genome as an ingroup (by adding it to a node or a branch), at most a single ancestral sequence is re-inferred. This prevents any information from the new genome from propagating to the rest of the tree. While this saves significant effort in recomputing other parts of the alignment, it also means that, occasionally, rare stretches of sequence in a newly added genome would not be properly aligned to distant outgroups because they were deleted or missing in the new genome's close relatives. Re-inferring the ancestral genomes on the path from newly added genomes to the root should address this issue if it appears.

We tested the effect of adding a new genome to an existing alignment using the same set of simulated catarrhine genomes as described above. To replicate the use-case of an end-user wanting to add a genome to a previously-created alignment, we generated an alignment holding out one of the 20 genomes (the crab-eating macaque), and added that genome back

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

into the alignment by both splitting an existing branch (resulting in the same topology as a full alignment would), and by adding the macaque as a new child of an existing ancestor (creating a trifurcation which did not exist in the original tree. All alignments for this analysis were generated using Progressive Cactus commit 49e80082 and we used tools from the hal package (https://github.com/ComparativeGenomicsToolkit/hal, commit 68db41d).

To add the crab-eating macaque back in as the child of an existing node (the add-to-node strategy), we ran a single new alignment with the tree (Rhesus:0.006, Crab_eating_macaque:0.007, Sooty_mangabey:0.001)anc6e;. The anc6e genome from the original, held-out alignment was used as an unreconstructed ancestral input sequence. We set the “runMapQFiltering” option in the config file to “0” and the “alignmentFilter” option to “singleCopyOutgroup”, since these options produce a better alignment of polytomies. We merged the resulting HAL file into a new copy of the existing alignment via the command:

```
halReplaceGenome <copy of held-out alignment> anc6e \
--topAlignmentFile <held-out alignment> \
--bottomAlignmentFile <add-to-node alignment>.
```

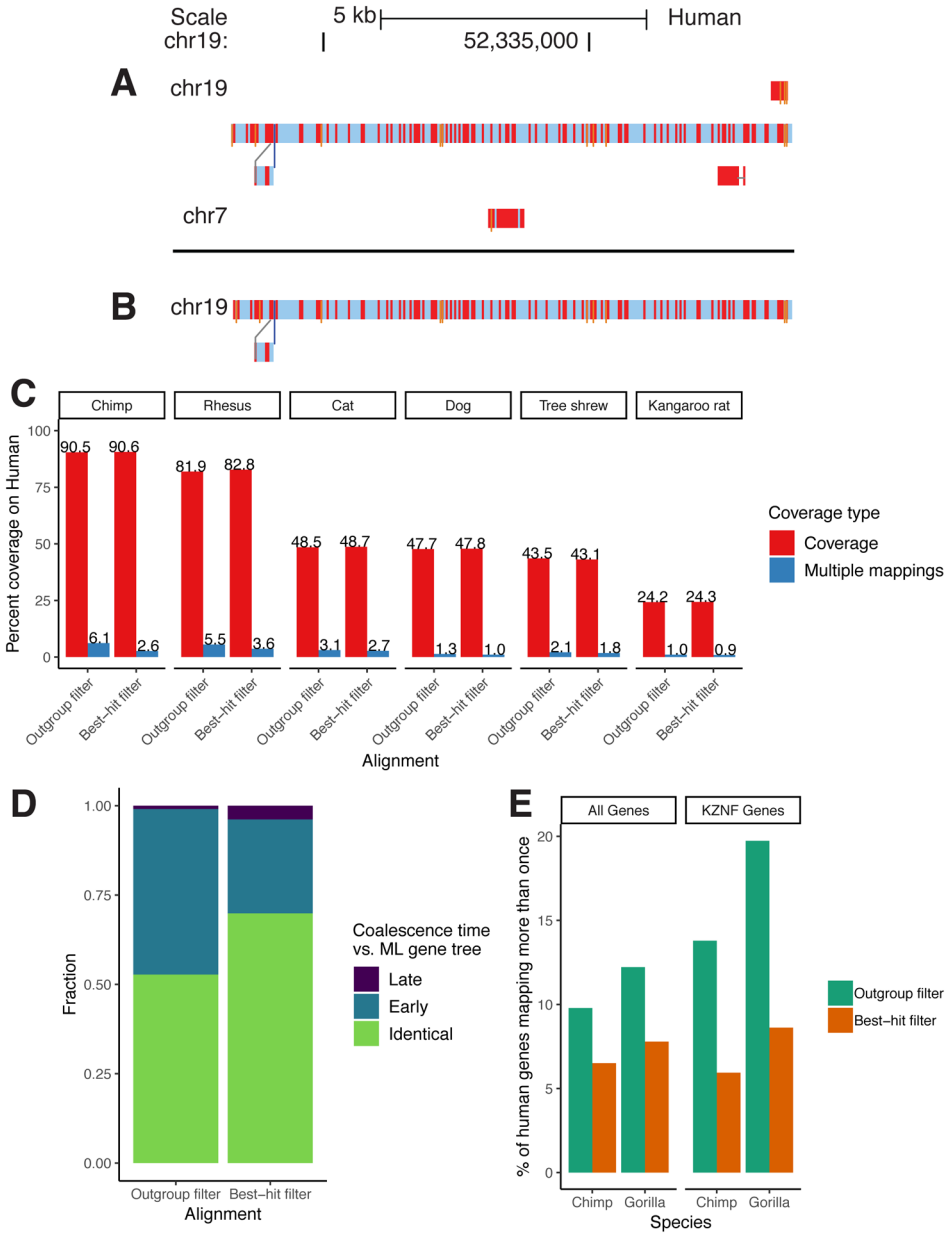
To add the macaque by splitting a branch (the add-to-branch strategy), we ran two separate alignments. We ran the first with the tree (((Rhesus:0.004991, Crab_eating_macaque:0.005991) anc5e:0.001, Sooty_mangabey:0.001)anc6e:0.005, Baboon:0.003042)anc7e; (with the --root anc5e option so that only a single subproblem was run), generating a newly reconstructed anc5e ancestor. We then ran a second alignment with the tree (anc5e:0.001, Sooty_mangabey:0.001)anc6e;, again providing the anc6e assembly from the original alignment rather than inferring a new reconstruction. (We note that these two subproblems could have been run in a single alignment invocation, resulting in the same amount of alignment work but a slightly more complicated merging process.) To merge these two add-to-branch intermediate alignments into a full alignment, we first removed the Rhesus genome from a new copy of the held-out alignment. We then ran halAddToBranch <held-out alignment> <first add-to-branch alignment> <second add-to-branch alignment> anc6e anc5e

```
Rhesus Crab_eating_macaque 0.001 0.006.
```

Both methods resulted in alignments that had accuracy deviating less than 0.1% from the full alignment that included the macaque from the start: both addition methods, as well as the full alignment, achieved an F1 score of 0.926 (Extended Data Table 1). We evaluated the performance of these new alignments using mafComparator in the same way as described above. In the interest of narrowly determining the accuracy of alignments involving the newly added genome, we counted only aligned pairs involving the Crab_eating_macaque genome when calculating precision, recall, and F1 scores.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

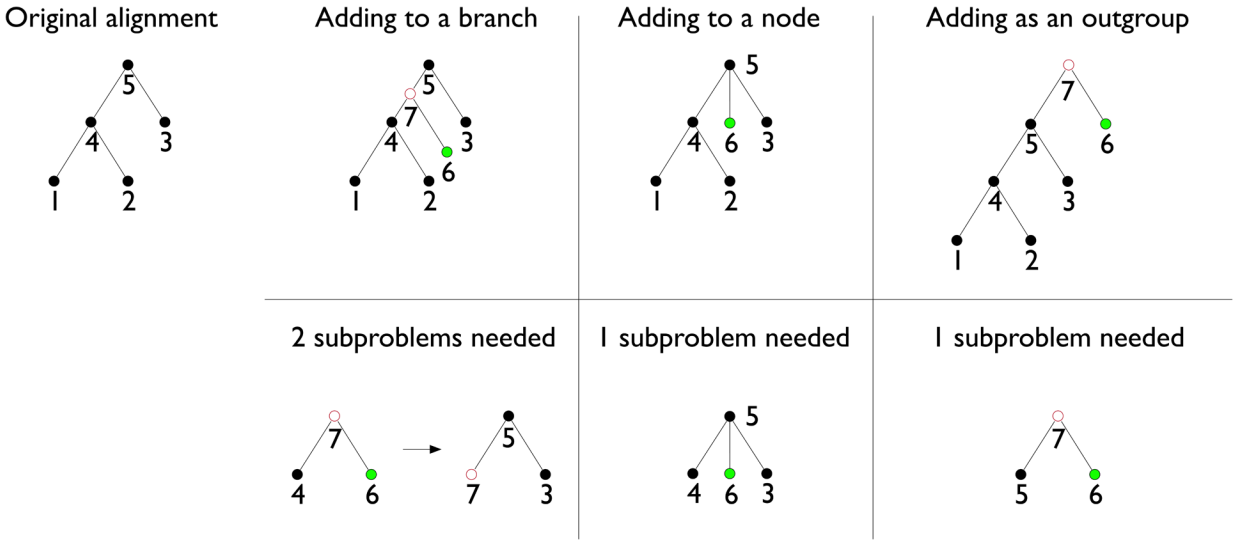
Extended Data



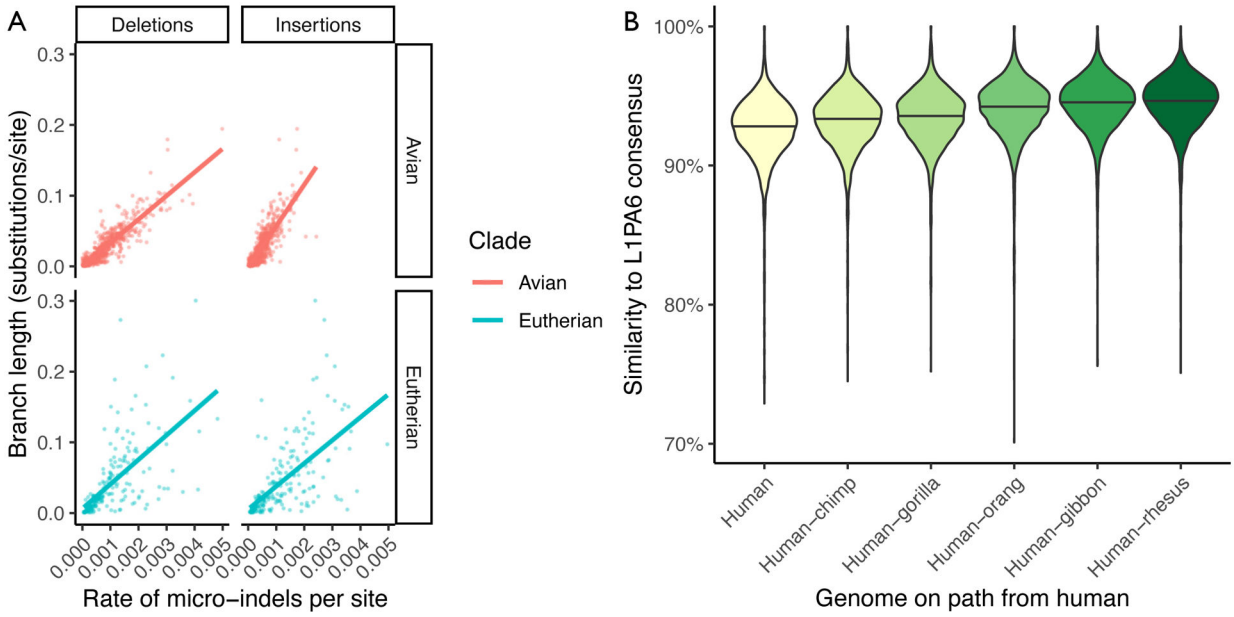
Extended Data Figure 1: Results from improved paralog-filtering.

A/B: A sample snake track [45] within a recently duplicated region before (A) and after (B) the filtering change. Nucleotide substitutions are shown as red bars, and insertions are shown as thin orange bars. C: Coverage results from two alignments of identical assemblies using the outgroup and best-hit filtering methods. Multiple-mappings: sites which map to two or more sites on the target genome. D: Results from comparing phylogenetic trees implicit in the HAL alignment to ML re-estimated trees of the same regions. “Early” coalescences imply that too many duplication events have been created in the reconciled trees, while “Late” implies that too many loss events have been created. E: Percent of human genes that map more than once to the chimp/gorilla genomes in two CAT [9] annotations using

alignments created with the outgroup and best-hit filtering methods. KZNF: KRAB zinc-finger genes.



Extended Data Figure 2: Methods of adding a genome to a Progressive Cactus alignment.
The top row shows the different ways of adding a new genome given its phylogenetic position, and the bottom row shows what subproblems would need to be computed for the new genome to be properly merged into the existing alignment. Green circles represent a new genome, and red circles represent newly reconstructed genomes.



Extended Data Figure 3: Analyzing insertions, deletions and L1PA6 repeats in the 600-way alignement.
A: Rates of micro-insertions and -deletions (micro-indels) along each branch within the 600-way, compared to conventional substitutions/site branch length. The data from avian and

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

eutherian branches are separated. Lines show a best-fit linear model for each category. B: Violin plot showing the increasing similarity to consensus of L1PA6 elements within reconstructed ancestral genomes along the path to the emergence of modern L1PA6 elements (in the human-rhesus ancestor). Horizontal lines indicate the median values.

Extended Data Table 1:
Adding a new genome to an alignment of simulated genomes.

Precision, recall, and F1-score statistics are all of the aligned pairs that contain a base of the added genome. An alignment where the genome was included initially is shown for comparison.

Alignment	Precision	Recall	F1-score	CPU time
Genome added to branch	97.27%	88.40%	92.63%	11 h
Genome added to node	97.21%	88.35%	92.57%	16 h
Full realignment of entire tree & new genome	97.19%	88.33%	92.55%	176 h

Extended Data Table 2:
Alignment similarity between four alignments of the same 48 avian genomes with different guide trees.

The similarity between each pair of alignments is represented by the F1 score of aligned-pair relationships in the two alignments.

Guide tree	Jarvis	Prum	Consensus	Permuted
Jarvis	1			
Prum	0.9867	1		
Consensus	0.9882	0.9883	1	
Permuted	0.9843	0.9822	0.9836	1

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

The research reported in this publication was supported by the National Institutes of Health under Award Numbers U01HG010961, U41HG010972, R01HG010485, 2U41HG007234, 5U54HG007990, 5T32HG008345-04, and U01HL137183. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. Parts of this work and its text were also included in Joel Armstrong's PhD thesis [46].

References

[1]. Eid J et al. Real-time DNA sequencing from single polymerase molecules. Science 323, 133–138 (2009). [PubMed: 19023044]

[2]. Weisenfeld NI, Kumar V, Shah P, Church DM & Jaffe DB Direct determination of diploid genome sequences. Genome Res. 27, 757–767 (2017). [PubMed: 28381613]

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

[3]. Jain M, Olsen HE, Paten B & Akeson M The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome Biol. 17, 239 (2016). [PubMed: 27887629]

[4]. Kitts PA et al. Assembly: a resource for assembled genomes at NCBI. Nucleic Acids Res. 44, 73–80 (2016).

[5]. Jain M et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat. Biotechnol 36, 338–345 (2018). [PubMed: 29431738]

[6]. Paten B et al. Cactus: Algorithms for genome multiple sequence alignment. Genome research 21, 1512–1528 (2011). [PubMed: 21665927]

[7]. Liu L, Yu L & Edwards SV A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evol. Biol 10, 302 (2010). [PubMed: 20937096]

[8]. Zhang C, Rabiee M, Sayyari E & Mirarab S ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. BMC Bioinformatics 19, 153 (2018). [PubMed: 29745866]

[9]. Fiddes IT et al. Comparative annotation toolkit (CAT)-simultaneous clade and personal genome annotation. bioRxiv 231118 (2017).

[10]. König S, Romoth LW, Gerischer L & Stanke M Simultaneous gene finding in multiple genomes. Bioinformatics 32, 3388–3395 (2016). [PubMed: 27466621]

[11]. Hubisz MJ, Pollard KS & Siepel A PHAST and RPHAST: Phylogenetic analysis with space/time models. Briefings in Bioinformatics 12, 41–51 (2011). [PubMed: 21278375]

[12]. Garber M et al. Identifying novel constrained elements by exploiting biased substitution patterns. Bioinformatics 25, 54–62 (2009). [PubMed: 18628288]

[13]. Armstrong J, Fiddes IT, Diekhans M & Paten B Whole-Genome Alignment and Comparative Annotation. Annu Rev Anim Biosci (2018).

[14]. Earl D et al. Alignathon: a competitive assessment of whole-genome alignment methods. Genome research 24, 2077–2089 (2014). [PubMed: 25273068]

[15]. Feng DF & Doolittle RF Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of molecular evolution 25, 351–360 (1987). [PubMed: 3118049]

[16]. Green RE et al. Three crocodilian genomes reveal ancestral patterns of evolution among archosaurs. Science 346, 1254449–1254449 (2014). URL http://www.sciencemag.org/cgi/doi/10.1126/science.1254449. 15334406. [PubMed: 25504731]

[17]. Dobrynin P et al. Genomic legacy of the African cheetah, Acinonyx jubatus. Genome Biol. 16, 277 (2015). [PubMed: 26653294]

[18]. Gordon D et al. Long-read sequence assembly of the gorilla genome. Science 352, aae0344 (2016). [PubMed: 27034376]

[19]. Lilue J et al. Sixteen diverse laboratory mouse reference genomes define strain-specific haplotypes and novel functional loci. Nat. Genet 50, 1574–1583 (2018). [PubMed: 30275530]

[20]. Kronenberg ZN et al. High-resolution comparative analysis of great ape genomes. Science 360 (2018).

[21]. Darling AE, Mau B & Perna NT Progressivemauve: Multiple genome alignment with gene gain, loss and rearrangement. PLoS ONE 5 (2010).

[22]. Blanchette M et al. Aligning multiple genomic sequences with the threaded blockset aligner. Genome Research 14, 708–715 (2004). [PubMed: 15060014]

[23]. Bray N & Pachter L MAVID: constrained ancestral alignment of multiple sequences. Genome Res. 14, 693–699 (2004). [PubMed: 15060012]

[24]. Harris R Improved pairwise alignment of genomic DNA. Ph.D. thesis, The Pennsylvania State University (2007).

[25]. Vivian J et al. Toil enables reproducible, open source, big biomedical data analyses. Nature biotechnology 35, 314 (2017).

[26]. Kurtzer GM, Sochat V & Bauer MW Singularity: Scientific containers for mobility of compute. PLoS ONE 12, 1–20 (2017).

[27]. Hickey G, Paten B, Earl D, Zerbino D & Haussler D Hal: a hierarchical format for storing and analyzing multiple genome alignments. Bioinformatics btt128 (2013).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

[28]. Edgar RC, Asimenos G, Batzoglou S & Sidow A Evolver: a whole-genome sequence evolution simulator. https://www.drive5.com/evolver.

[29]. Prum RO et al. A comprehensive phylogeny of birds (Aves) using targeted next-generation DNA sequencing. Nature 526, 569–573 (2015). [PubMed: 26444237]

[30]. Jarvis ED et al. Whole-genome analyses resolve early branches in the tree of life of modern birds. Science 346, 1320–1331 (2014). [PubMed: 25504713]

[31]. Kent WJ, Baertsch R, Hinrichs A, Miller W & Haussler D Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. Proceedings of the National Academy of Sciences 100, 11484–11489 (2003).

[32]. Genereux D et al. Genomics in an age of extinction (in submission).

[33]. Zhang G et al. Genomics: Bird sequencing project takes off. Nature 522, 34 (2015).

[34]. Zhang G et al. Comparative genomics reveals insights into avian genome evolution and adaptation. Science 346, 1311–1320 (2014). [PubMed: 25504712]

[35]. Chen J-Q et al. Variation in the Ratio of Nucleotide Substitution and Indel Rates across Genomes in Mammals and Bacteria. Molecular Biology and Evolution 26, 1523–1531 (2009). URL 10.1093/molbev/msp063. http://oup.prod.sis.lan/mbe/article-pdf/26/7/1523/13640777/msp063.pdf. [PubMed: 19329651]

[36]. Smit AFA and Hubley R and Green P RepeatMasker Open-4.0 http://www.repeatmasker.org (2013–2015).

[37]. Kent WJ Blat—the blast-like alignment tool. Genome Research 12, 656–664 (2002). URL http://genome.cshlp.org/content/12/4/656.abstract. http://genome.cshlp.org/content/12/4/656.full.pdf+html. [PubMed: 11932250]

[38]. Camacho C et al. Blast+: architecture and applications. BMC Bioinformatics 10, 421 (2009). URL 10.1186/1471-2105-10-421. [PubMed: 20003500]

[39]. Rhie A et al. Towards complete and error-free genome assemblies of all vertebrate species. bioRxiv (2020). URL https://www.biorxiv.org/content/early/2020/05/23/2020.05.22.110833.

[40]. Koepfli K-P, Paten B, the Genome 10K Community of Scientists & O'Brien, S. J. The genome 10k project: A way forward. Annual Review of Animal Biosciences 3, 57–111 (2015). URL 10.1146/annurev-animal-090414-014900., 10.1146/annurev-animal-090414-014900. [PubMed: 25689317]

[41]. Lewin HA et al. Earth biogenome project: Sequencing life for the future of life. Proceedings of the National Academy of Sciences 115, 4325–4333 (2018). URL https://www.pnas.org/content/115/17/4325. https://www.pnas.org/content/115/17/4325.full.pdf.

[42]. Shafin K et al. Nanopore sequencing and the shasta toolkit enable efficient de novo assembly of eleven human genomes. Nature Biotechnology (2020). URL 10.1038/s41587-020-0503-6.

[43]. Hickey G et al. Genotyping structural variants in pangenome graphs using the vg toolkit. Genome Biol. 21, 35 (2020). [PubMed: 32051000]

[44]. Sherman RM et al. Assembly of a pan-genome from deep sequencing of 910 humans of African descent. Nat. Genet 51, 30–35 (2019). [PubMed: 30455414]

[45]. Nguyen N et al. Comparative assembly hubs: Web-accessible browsers for comparative genomics. Bioinformatics 30, 3293–3301 (2014). arXiv:1311.1241v1. [PubMed: 25138168]

[46]. Armstrong J Enabling comparative genomics at the scale of hundreds of species. Ph.D. thesis, University of California Santa Cruz (2019). URL 10.5281/zenodo.3936004.

[47]. Paradis E, Claude J & Strimmer K APE: Analyses of Phylogenetics and Evolution in R language. Bioinformatics 20, 289–290 (2004). URL 10.1093/bioinformatics/btg412. http://oup.prod.sis.lan/bioinformatics/article-pdf/20/2/289/533578/btg412.pdf. [PubMed: 14734327]

[48]. Revell LJ phytools: an r package for phylogenetic comparative biology (and other things). Methods in Ecology and Evolution 3, 217–223 (2012). URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.2041-210X.2011.00169.x. https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2041-210X.2011.00169.x.

[49]. Bao W, Kojima KK & Kohany O Repbase Update, a database of repetitive elements in eukaryotic genomes. Mob DNA 6, 11 (2015). [PubMed: 26045719]

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

[50]. Kumar S, Stecher G, Suleski M & Hedges SB TimeTree: A Resource for Timelines, Timetrees, and Divergence Times. Molecular Biology and Evolution 34, 1812–1819 (2017). URL 10.1093/molbev/msx116. http://oup.prod.sis.lan/mbe/article-pdf/34/7/1812/17653096/msx116.pdf. [PubMed: 28387841]

[51]. Felsenstein J PHYLIP: Phylogeny Inference Package (Version 3.2). Cladistics 5, 164–166 (1989).

[52]. Haeussler M et al. The UCSC Genome Browser database: 2019 update. Nucleic Acids Res. 47, D853–D858 (2019). [PubMed: 30407534]

[53]. Ondov BD et al. Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol. 17, 132 (2016). [PubMed: 27323842]

[54]. Zhu J et al. Comparative genomics search for losses of long-established genes on the human lineage. PLoS Computational Biology 3, e247 EP – (2007). URL 10.1371/journal.pcbi.0030247. [PubMed: 18085818]

[55]. Chiaromonte F, Yap VB & Miller W Scoring pairwise genomic sequence alignments. Biocomputing 2002 (2001). URL 10.1142/9789812799623_0012.

[56]. Schwartz S Human-mouse alignments with blastz. Genome Research 13, 103–107 (2003). URL 10.1101/gr.809403. [PubMed: 12529312]

[57]. Pevzner PA, Pevzner PA, Tang H & Tesler G De novo repeat classification and fragment assembly. Genome Res. 14, 1786–1796 (2004). [PubMed: 15342561]

[58]. Medvedev P & Brudno M Maximum likelihood genome assembly. J. Comput. Biol 16, 1101–1116 (2009). [PubMed: 19645596]

[59]. Paten B et al. Superbubbles, Ultrabubbles, and Cacti. J. Comput. Biol 25, 649–663 (2018). [PubMed: 29461862]

[60]. Huntley S et al. A comprehensive catalog of human KRAB-associated zinc finger genes: insights into the evolutionary history of a large family of transcriptional repressors. Genome Res. 16, 669–677 (2006). [PubMed: 16606702]

[61]. Price MN, Dehal PS & Arkin AP FastTree 2–approximately maximum-likelihood trees for large alignments. PLoS ONE 5, e9490 (2010). [PubMed: 20224823]

[62]. Frankish A et al. Gencode reference annotation for the human and mouse genomes. Nucleic acids research 47, D766–D773 (2018).

[63]. Paten B et al. Cactus graphs for genome comparisons. J. Comput. Biol 18, 469–481 (2011). [PubMed: 21385048]

[64]. Nguyen N et al. Building a pan-genome reference for a population. Journal of computational biology : a journal of computational molecular cell biology 22, 387–401 (2015). URL http://online.liebertpub.com/doi/10.1089/cmb.2014.0146. [PubMed: 25565268]

[65]. Jukes TH & Cantor CR Evolution of protein molecules. Mammalian protein metabolism 3, 132 (1969).

[66]. Felsenstein J Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. Systematic Biology 22, 240–249 (1973). URL 10.1093/sysbio/22.3.240. http://oup.prod.sis.lan/sysbio/article-pdf/22/3/240/4741566/22-3-240.pdf.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

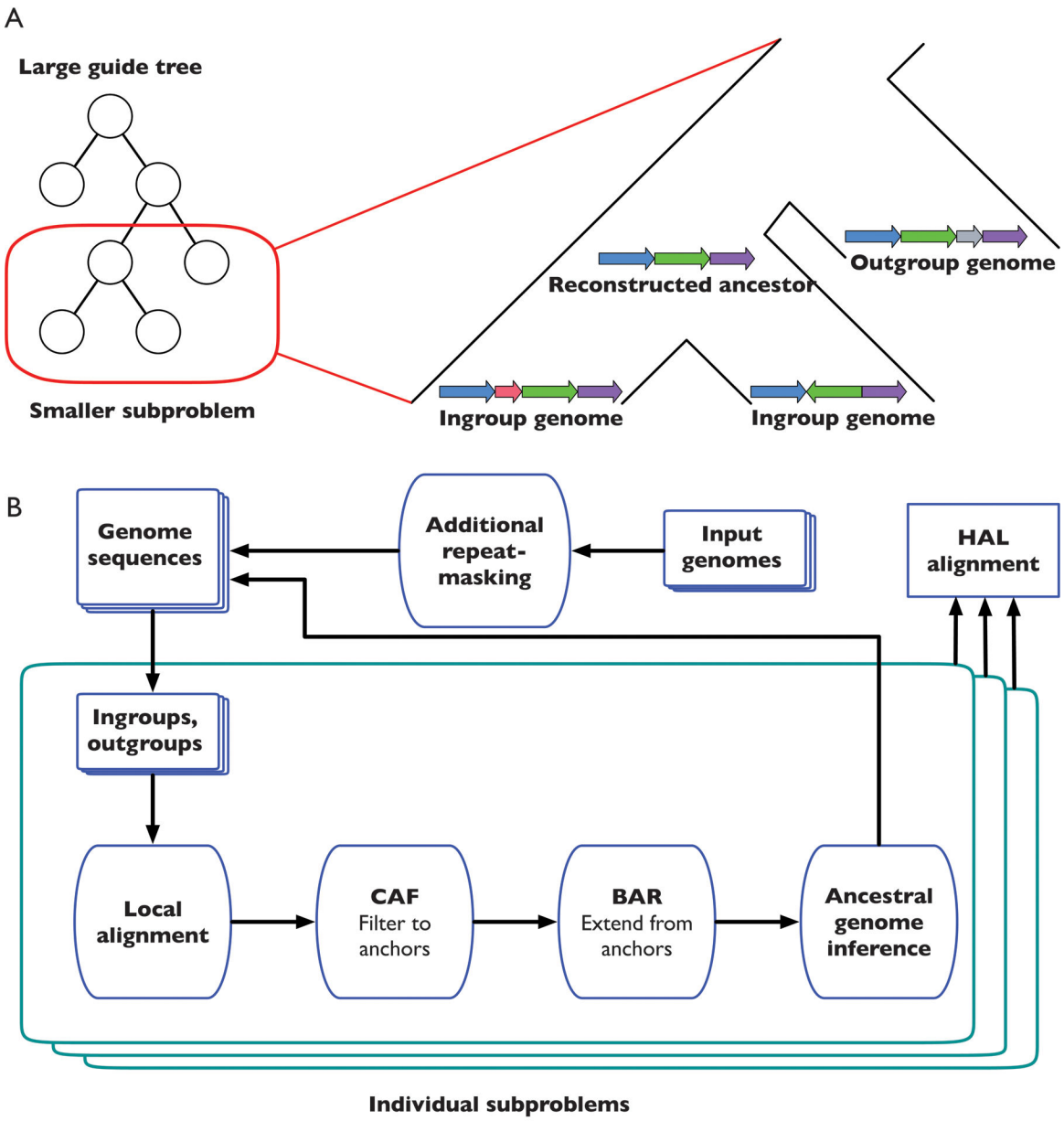


Figure 1: The alignment process within Progressive Cactus.

A: A large alignment problem is decomposed into many smaller subproblems using an input guide tree. Each subproblem compares a set of ingroup genomes (the children of the internal node to be reconstructed) against each other as well as a sample of outgroup genomes (non-descendants of the internal node in question). B: This flowchart represents the phases in which the overall alignment, as well as each subproblem alignment, proceeds through. The end result is a new genome assembly representing Progressive Cactus's reconstruction of the ancestral genome, as well as an alignment between this ancestral genome and its children. After all subproblems have been completed, the parent-child alignments are combined to create the full reference-free alignment in the HAL [27] format.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

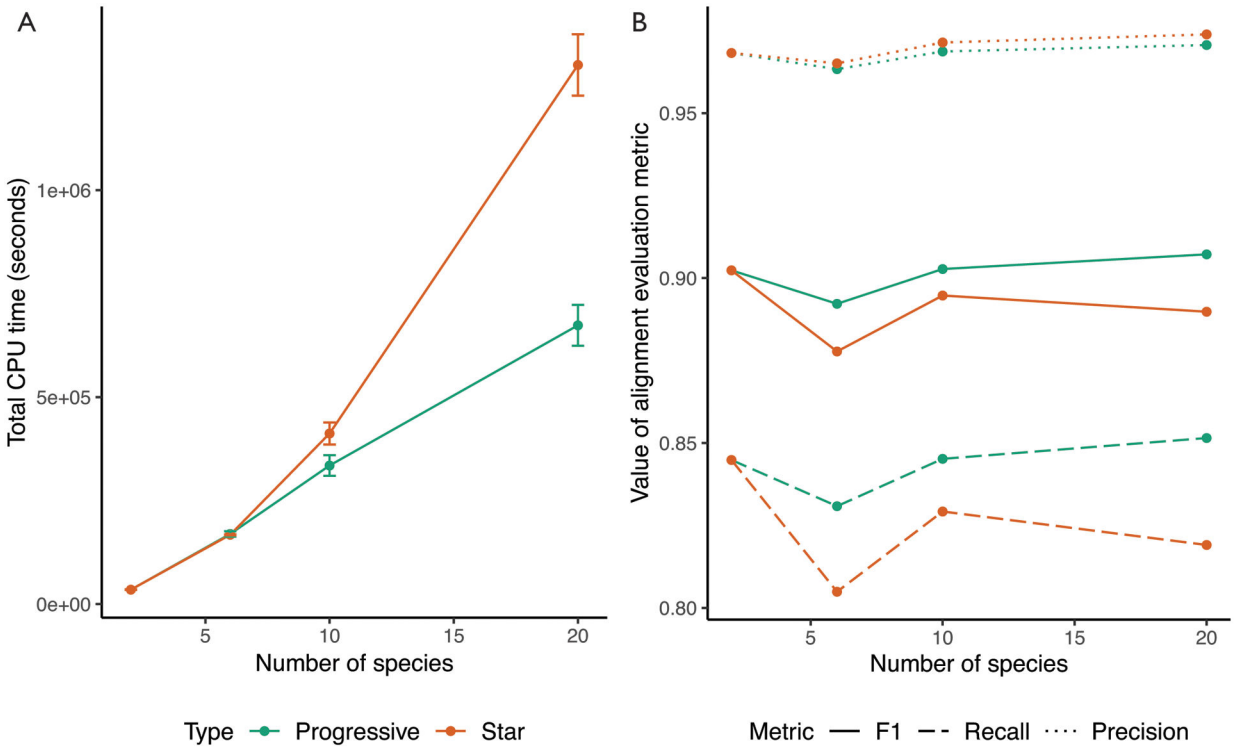


Figure 2: Comparing alignments of varying numbers of simulated genomes using Progressive Cactus.

The progressive mode of Progressive Cactus ("Progressive") is shown, versus the mode without progressive decomposition that is similar to that originally described in [6] ("Star"). A: The average total runtime of the two alignment methods across 3 runs. Error bars show the standard deviation of runtime between the runs. The runtime is identical when aligning two genomes since the alignment problem is not further decomposed, but the linear scaling of the progressive mode means it is much faster with large numbers of genomes than the quadratic scaling required without progressive alignment. B: The precision, recall, and F1 score (harmonic mean of precision and recall) of aligned pairs for each alignment compared to pairs from the true alignment produced by the simulation.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

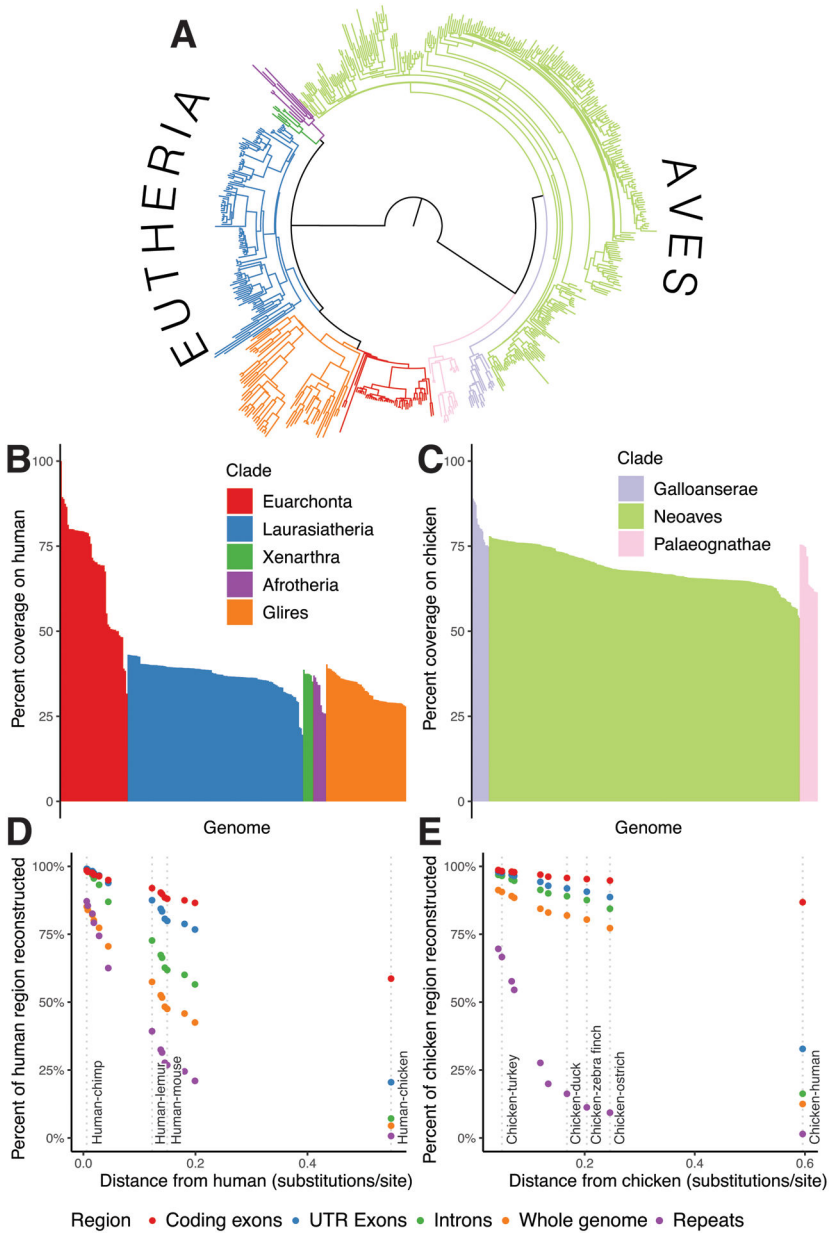


Figure 3: Analyzing the 600-way amniote alignment.

A: The species tree relating the 600 genomes. Branches are colored by clades in the same way as figures B and C. B: Percent coverage on human within the eutherian mammals, grouped by clade from highest to lowest coverage. C: Similar to B, but for coverage on chicken within the avian alignment. D: Percent of various regions within the human genome mappable to each ancestral genome reconstructed along the path leading from human to the root. The positions of selected ancestors are labeled by dotted lines to indicate useful taxonomic reference points as context. E: Similar to D, but for the path of reconstructed ancestors between chicken and the root.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

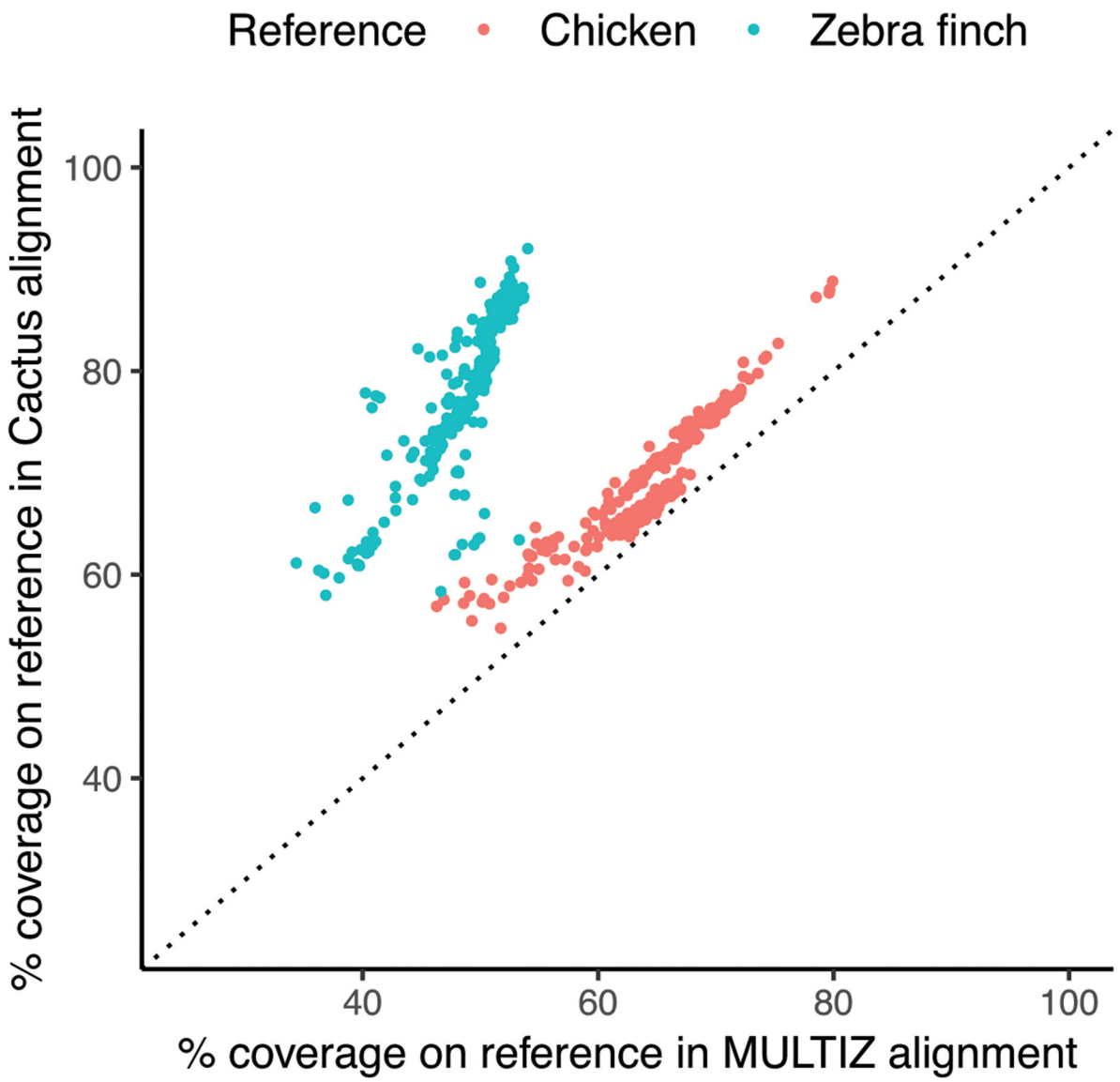


Figure 4: Comparing Cactus and Multiz alignment coverage.

A comparison of coverage in the Progressive Cactus avian alignment compared to a chicken-referenced MULTIZ [22] alignment of the same genomes. Coverage of both alignments on chicken and zebra finch is shown to illustrate the effects of reference-bias on the completeness of the MULTIZ alignment. The diagonal dotted line indicates a slope of 1 (i.e. if MULTIZ and Progressive Cactus coverage were equal).

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 1:
Aggregate statistics for the 600-way alignment.

The increase in computational work for the mammal alignment over the bird alignment is largely caused by the increase in the pairwise alignment phase runtime because it scales quadratically with the size of the genomes being aligned.

Alignment	# of genomes	Total bases	Instance-hours	Core-hours	Common ancestor size
Zoonomia	242	669 billion	68,166	1.9 million	1.73 Gb
Bird 10K	363	400 billion	5,302	0.2 million	1.13 Gb
Combined	605	1.07 trillion	73,692	2.1 million	181 Mb