BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 23 2014, pages 3293–3301
doi:10.1093/bioinformatics/btu534

Genome analysis

Advance Access publication August 18, 2014

# Comparative assembly hubs: Web-accessible browsers for comparative genomics

Ngan Nguyen[1,†], Glenn Hickey[1,†], Brian J. Raney[1,†], Joel Armstrong[1], Hiram Clawson[1], Ann Zweig[1], Donna Karolchik[1], William James Kent[1], David Haussler[1,2] and Benedict Paten[1,*]

[1]Center for Biomolecular Sciences and Engineering, CBSE/ITI, UC Santa Cruz, 1156 High St, Santa Cruz, CA 95064, USA and [2]Howard Hughes Medical Institute, Center for Biomolecular Science and Engineering, UCSC, 1156 High Street, Santa Cruz, CA 95064, USA

Associate Editor: John Hancock

**ABSTRACT**

**Motivation:** Researchers now have access to large volumes of genome sequences for comparative analysis, some generated by the plethora of public sequencing projects and, increasingly, from individual efforts. It is not possible, or necessarily desirable, that the public genome browsers attempt to curate all these data. Instead, a wealth of powerful tools is emerging to empower users to create their own visualizations and browsers.

**Results:** We introduce a pipeline to easily generate collections of Web-accessible UCSC Genome Browsers interrelated by an alignment. It is intended to democratize our comparative genomic browser resources, serving the broad and growing community of evolutionary genomicists and facilitating easy public sharing via the Internet. Using the alignment, all annotations and the alignment itself can be efficiently viewed with reference to any genome in the collection, symmetrically. A new, intelligently scaled alignment display makes it simple to view all changes between the genomes at all levels of resolution, from substitutions to complex structural rearrangements, including duplications. To demonstrate this work, we create a comparative assembly hub containing 57 *Escherichia coli* and 9 *Shigella* genomes and show examples that highlight their unique biology.

**Availability and implementation:** The source code is available as open source at: https://github.com/glennhickey/progressiveCactus The *E.coli* and *Shigella* genome hub is now a public hub listed on the UCSC browser public hubs Web page.

**Contact:** benedict@soe.ucsc.edu

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

Received on April 9, 2014; revised on July 31, 2014; accepted on August 1, 2014

## 1 INTRODUCTION

Visualization is key to understanding functional and comparative genomic information. Genome browsers are therefore critical to the study of biology, providing accessible resources for displaying annotations and alignments. The UCSC Genome Browser (Karolchik *et al*. 2014) is one of the most popular, but creating a reference genome browser for a new genome within it previously required significant resources, as it was necessary to create a mirror site to host the new reference browser separately, or to work with the staff of the genome browser to create a new reference browser within their main site. Assembly hubs (Karolchik *et al*. 2014), which build on the successful track hub model (Raney *et al*. 2013), make it easy to generate an individual UCSC browser simply by hosting the data in the form of flat files on any publicly addressable URL. This avoids users having to install and configure the substantial browser code base on their machines and, by using user hosting, makes updating straightforward. However, with access to low-cost sequencing technology and the wealth of genomes already available, users increasingly want to be able to create not only a single custom browser, but also sets of genome browsers. This work is intended to meet that growing need; it extends assembly hubs to allow users to quickly create 'comparative assembly hubs', a framework of multiple genome browsers and annotations interrelated by an alignment. As part of this development, we introduce a series of novel features intended to improve visualization, exploration and community sharing of novel comparative genomics data.

Displaying multiple genome alignments is extremely challenging owing to both the high dimensionality and volume of the underlying data [see Nielsen *et al*. (2010) for a review]. There are currently three main ways to visualize multiple genome alignments: dot plots [e.g. DAGChainer (Haas *et al*. 2004), VISTA-Dot (Mayor *et al*. 2000), etc.], circle plots [e.g. Circos (Krzywinski *et al*. 2009), GenomeRing (Herbig *et al*. 2012), etc.] and linear, row-oriented representations [e.g. VISTA (Mayor *et al*. 2000), Jalview (Waterhouse *et al*. 2009), IGV (Thorvaldsdóttir *et al*. 2013), etc.]. Each of these visualization methods has benefits and weaknesses. Dots plots, having two dimensions, provide equivalently powerful representations of two genomes in one graphic; however, they are pairwise and therefore unsuitable for the display of multiple genome alignments. Circular genome plots are typically less visually cluttered than linear representations for viewing genomic rearrangements, but are less useful for the display of global multiple sequence alignments, as might be shown at the gene or base level. Linear representations have the advantage that they fit neatly with the genome browser displays and tracks, and are flexible, in that they work reasonably at multiple levels of resolution.

Here we introduce a new linear display—the snake track—that, to our knowledge, is the first linear representation to

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first three authors should be regarded as Joint First Authors.

allow the viewing of all variations, including structural rearrangements, duplications, substitutions, insertions and deletions in a single, conceptually simple, interactive visualization. In addition, by using a novel algorithm to generate procedural levels of detail (LOD), it is viewable at every resolution, from complete chromosomes to individual residues. Being based on a symmetric reference independent alignment format (Hickey *et al*. 2013), for the first time for an alignment view in the UCSC browser, snake tracks are viewable between any set of genomes in the hub and from any chosen genome, as all genomes in a hub have a generated reference browser. This is important because restricting browsing to a single reference can limit visualization of regions in other genomes that do not map to the reference, or which are rearranged with respect to a reference. Finally, to overcome the limitation of viewing the alignment only from the perspective of any single reference genome, each comparative assembly hub provides an automatically generated pangenome reference browser, using our recently developed algorithm (Nguyen *et al*. 2014). Similar to the pioneering visualization concepts described in Herbig *et al*. (2012), this pangenome includes a single consensus copy of every set of homologous subsequences in an ordering that as closely as possible reflects the (weighted) consensus of the input genomes. We will show how this can be used to create unique views of the data—here we demonstrate the creation of an *Escherichia coli* core genome.

Besides a new visualization, the comparative assembly hubs are novel from a genome browser perspective. The underlying alignment is used, by a process of lift-over (Zhu et al. 2007), to automatically project annotations to any genome in the hub, even if the annotations were originally mapped to just one genome. Previously, lift-over was used on a case-by-case basis within the genome browser to project tracks between assemblies, for example, when moving to an updated assembly. Here we make it a default, integral feature, making it easy to view putative genes and functional annotations on novel genomes by a process of translation through the underlying alignment, rather than through a set of (frequently inconsistent) *ad hoc* pairwise alignments. The comparative assembly hubs framework is also, to our knowledge, the first Web-based genome browser that allows easy public sharing of comparative data without requiring other users to download data—only a Web browser is required. Finally, separately to the novel features introduced, we have worked to integrate these user-generated browsers with many of the existing tools of the UCSC browser.

## 2 RESULTS

We first describe the software pipeline for building comparative assembly hubs, before describing them by example using an alignment of *E.coli* and *Shigella* genomes.

### 2.1 Comparative assembly hubs pipeline

Our software pipeline comprises three main modular components, for which we provide an overall distribution that can be downloaded and installed with a few commands (see Section 3), and which should work on unix distributions such as Linux, BSD and OS X. The first component is the Cactus alignment program (Paten *et al*. 2011), which takes as input a set of genome sequences and outputs a genome multiple sequence alignment in hierarchical alignment (HAL) format (Hickey *et al*. 2013). The second component is HAL tools, to which we have added a series of command-line tools and C/C++ Application Programming Interfaces (APIs) for manipulating HAL files and building comparative assembly hubs. The final component is the snake track display, which is now part of the UCSC Genome Browser code base (Karolchik *et al*. 2014), and which provides visualization of alignments directly from HAL files.

The pipeline is run in three steps. First, either Cactus is run to generate the HAL alignment file directly, or an MAF file, generated separately by an aligner such as Multiz (Miller *et al*. 2007), is converted into a HAL file using the maf2Hal tool (in the HAL tools package). Second, the hal2AssemblyHub script (in the HAL tools package) builds the comparative assembly hub using the HAL file and any set of annotation files provided, either in BED or WIG format (http://genome.ucsc.edu/FAQ/FAQformat.html). This script takes care of converting the base annotation files into the display scaleable bigBed and bigWig formats and translates these annotations (optionally) via a process of alignment lift-over (Zhu *et al*. 2007) to all the other genomes. At the end of this process, a directory is created that contains the necessary files, using compressed formats for minimal space usage. In the final step, the location of the 'hub.txt' file, addressable as a public URL, is pasted into the UCSC browser hub page to view the browsers.

The pipeline builds one browser for each input genome and, in addition, any ancestral or pangenomes that were imputed (e.g. by Cactus, if used) during the alignment process (Nguyen *et al*. 2014).

### 2.2 *Escherichia coli* comparative assembly hub

To demonstrate this work, we use a set of 57 *E.coli* and 9 *Shigella* spp. complete genomes. *Escherichia coli* contains substantial intraspecies genomic diversity, which allows for its high versatility and variation, encompassing various pathotypes, antibiotic resistances and lifestyles [see review in Leimbach *et al*. (2013)]. Comparative genomic analyses of multiple strains of *E.coli* have proven useful in understanding the molecular basis of their phenotypic differences and assisting in practical applications such as diagnostic and antibiotic developments for infectious disease (Didelot *et al*. 2012). Owing to intense study, *E.coli* is one of the most sequenced bacterial species, with the second highest number of complete sequenced genomes (after *Salmonella enterica*, source: http://www.ebi.ac.uk/genomes/bacteria.html) available at the time of writing. Automated tools for comparison and visualization are therefore critical for research efforts to keep pace with available data.

To illustrate a browser, Figure 1 displays a region of one of the *E.coli* reference genomes, K12 MG1655. The topmost tracks are K12 MG1655 annotations, including conservation, GC percentage, genes, antibiotic resistance database, genomic islands and non-coding RNAs (ncRNAs; rRNA and tRNA). Below these are snake tracks, showing the alignment of the genome to a subset of the other genomes, and a lifted-over ncRNA annotation track (track K12 W3110 RNA) of *E.coli* K12 W3110.

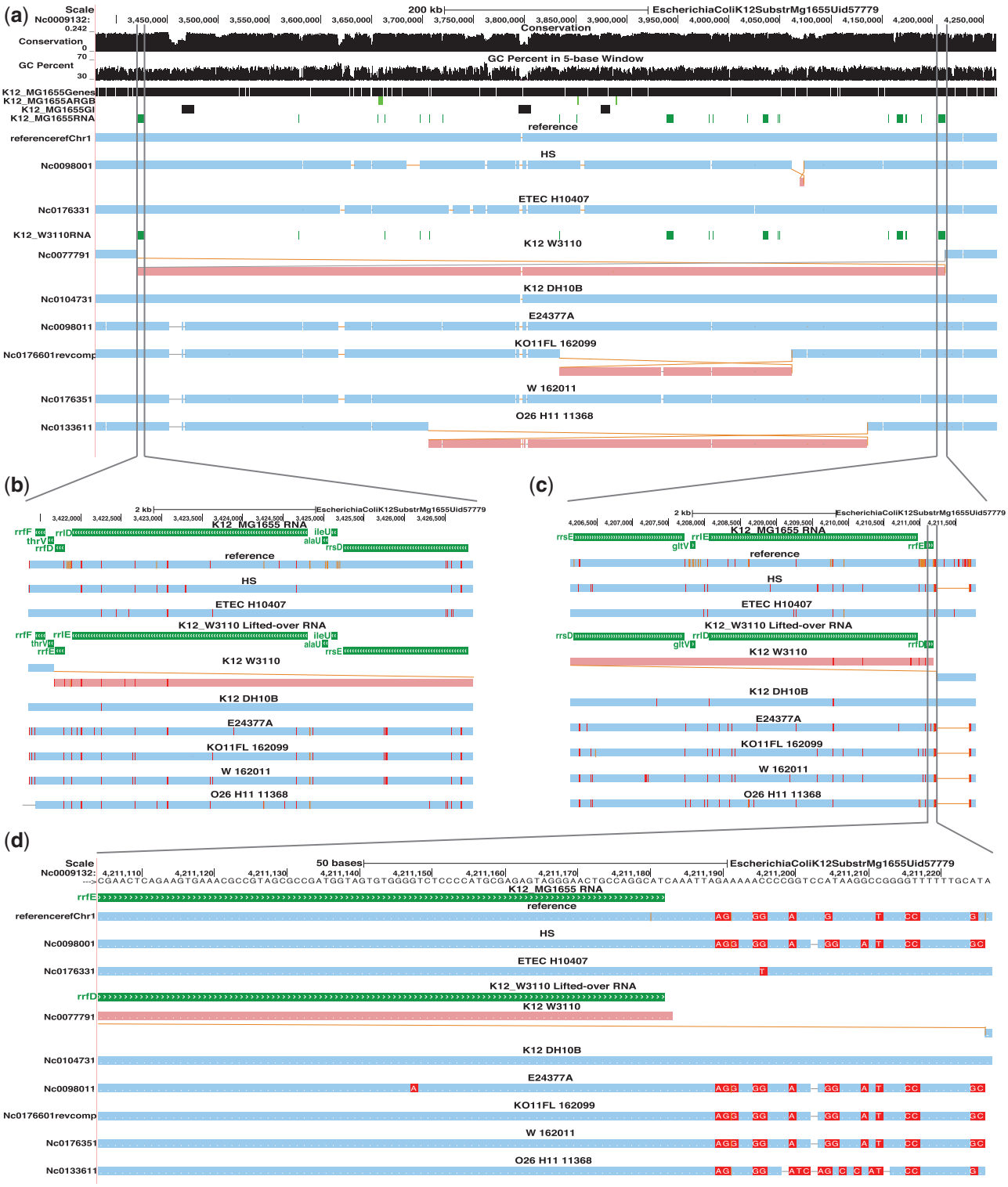


**Fig. 1.** An example *E.coli* comparative assembly hub with *E.coli* K12 MG1655 as the reference browser. The top browser screenshot (a) shows an ~900 kb region with a known large inversion (light red) in the closely related strain K12 W3110, which is flanked by homologous (with opposite orientations) ribosomal RNA operons *rrnD* and *rrnE* [Hayashi *et al.* (2006); Hill and Harnish (1981)], and is the result of recombination between them. (b–c) Zoom-in of the K12 W3110 inversion left and right boundaries, respectively, showing operon *rrnE* of K12 W3110 ('K12_W3110 RNA' track, in green, which is K12 W3110 ncRNA annotation track lifted-over to K12 MG1655) aligned to operon *rrnD* of K12 MG1655 ('K12_MG1655 RNA' track, also in green) on the left and operon *rrnD* of K12 W3110 aligned to operon *rrnE* of K12 MG1655 on the right. If further zoomed in (d), SNPs and query insertions are visible. The text on the screenshots was adjusted for better readability

*2.2.1 The snake track* Each snake track shows the relationship between the chosen browser genome, termed the reference (genome), and another genome, termed the query (genome). The snake display is capable of showing all possible types of structural rearrangement. When stacked together, snake tracks allow a flexible view of multiple genomes.

In *full* display mode (snake tracks in Fig. 1), a snake track can be decomposed into two primitive drawing elements: segments, which are the colored rectangles, and adjacencies, which are the lines connecting the segments. Segments represent subsequences of the query genome aligned to the given portion of the reference genome. Adjacencies represent the covalent bonds between the aligned subsequences of the query genome. Segments can be configured to be colored by chromosome, strand (as shown) or be left a single color. Layout of the segments is described in Section 3. At the highest LOD above that showing individual bases (i.e. the first level of summary detail above the original alignment), red tick marks within segments represent substitutions with respect to the reference (Fig. 1b and c). Further zooming in to the base level, these substitutions are labeled with the non-reference base (Fig. 1d). An insertion in the reference relative to the query creates a gap between abutting segment sides that is connected by an adjacency. An insertion in the query relative to the reference is represented either by an orange tick mark that splits a segment at the location the extra bases would be inserted or by coloring an adjacency orange, indicating that there are unaligned bases between the two connected segment ends.

More complex structural rearrangements create adjacencies that connect the sides of non-abutting segments in a natural fashion, e.g. Figure 1 shows the known large inversion in K12 MG1655's closely related strain K12 W3110 (Hill and Harnish 1981). The inversion is a result of a homologous recombination, with operon *rrnD*, consisting of *rrsD*, *ileU*, *alaU*, *rrlD*, *rrfD*, *thrV* and *rrfF*, and operon *rrnE*, consisting of *rrsE*, *gltV*, *rrlE* and *rrfE*, being homologous segments with opposite directions (Fig. 1b and c). Also shown in Figure 1 are two smaller inversions in KO11FL (Turner *et al.* 2012) and O26 H111 1368 (Ogura *et al.* 2009) and a relatively much smaller inversion in HS.

Duplications within the query genome create extra segments that overlap along the reference genome axis, e.g. Figure 2 shows a tandem repeat region of *E.coli* KO11FL 162099 displayed along the genome of *E.coli* KO11FL 52593 that was engineered by the chromosomal insertion of the *Zymomonas mobilis pdc*, *adhB* and *cat* genes into *E.coli* W for ethanol production purposes (Ohta *et al.* 1991); 162099 is a derivative of 52593 and contains 20 tandem copies of the inserted *pdc-adhB-cat* genes (Turner *et al.* 2012).

To show regions where the query segments align to multiple locations within the reference, for each snake track we draw color-coded sets of lines along the reference genome axis that indicate self-homologies (intervals of the reference genome that align to other intervals of the reference genome). To maintain the semantics of the snake, in which each query segment is drawn only once, we align any query segment that aligns to multiple segments in the reference to just the leftmost copy in the reference (Fig. 2), because otherwise the display ceases to be an alternating sequence of blocks and adjacencies representing the query. However, the multiple alignment information is not lost, as the overlap with the color-coded segments on the reference axis informs the viewer to interpret such a query segment as being multi-aligned to the reference.

The *pack* display option can be used to display a snake track in more limited vertical space. It eliminates the adjacencies from the display, and forces the segments onto as few rows as possible given the constraint of still showing duplications in the query sequence (e.g. track W 162101 of Fig. 2). The *dense* display further eliminates these duplications so that a snake track is compactly represented along just one row, essentially showing the coverage of the query genome on the reference (e.g. the last snake track of Fig. 2). Finally, the *squish* display is similar to *dense* but with a thinner thickness of the row to maximally reduce the vertical space (e.g track IAI1 of Fig. 2).

Clicking on segments moves the display to the corresponding region in the query genome, making it simple to navigate between references, all of which have equivalent displays. This symmetry frees the user from investigating the alignment from just one perspective. Various mouse-overs are implemented to show the sizes of display elements, and the snakes and annotations can be reordered by dragging.

*2.2.2 Procedural levels of alignment detail* The different LODs displayed in Figure 1a–d show the alignment at mega-base, kilobase and base levels. To achieve this in a Web browser, serving data across the Internet (generally still a relatively slow and high latency connection compared with keeping the data locally), we needed a novel solution, because, for instance, a chromosome will typically be decomposed into millions of segments in a HAL graph. In Section 3, we describe pre-generating interpolated HAL graphs that store only as much information as is visible on the screen at different zoom levels, and demonstrate that we achieve constant load times for Web pages at all levels of resolution using the method.

*2.2.3 Managing alignments and lifted annotations* A unique feature of comparative assembly hubs is that all annotations can be viewed, through the alignment, from any genome. To make managing the large number of possible snake and lifted-over annotations easy, for each browser, a central configuration page is provided that uses a grid layout as its basis (Fig. 3). This configuration page layout is adapted from the UCSC Encode Browser [Rosenbloom *et al.* (2010)] where, instead of using it to select tracks from combinations of cell line and assay types, it is instead used to select from the combination of genomes and (lifted-over) tracks, laid out phylogenetically (if a tree is provided). As with the Encode browser, the grid is sufficiently compact to display hundreds of tracks on a page.

*2.2.4 UCSC browser integration* A key benefit of comparative assembly hubs is their integration with the popular UCSC browser and the tools it provides. For example, export of subregions of the alignment and track intersections can be made via the UCSC table browser (Karolchik *et al.* 2004), and, via user sessions, individual browser displays can be shared (see the supplement for links to examples). Users may also add additional tracks using the 'Custom Tracks' function.

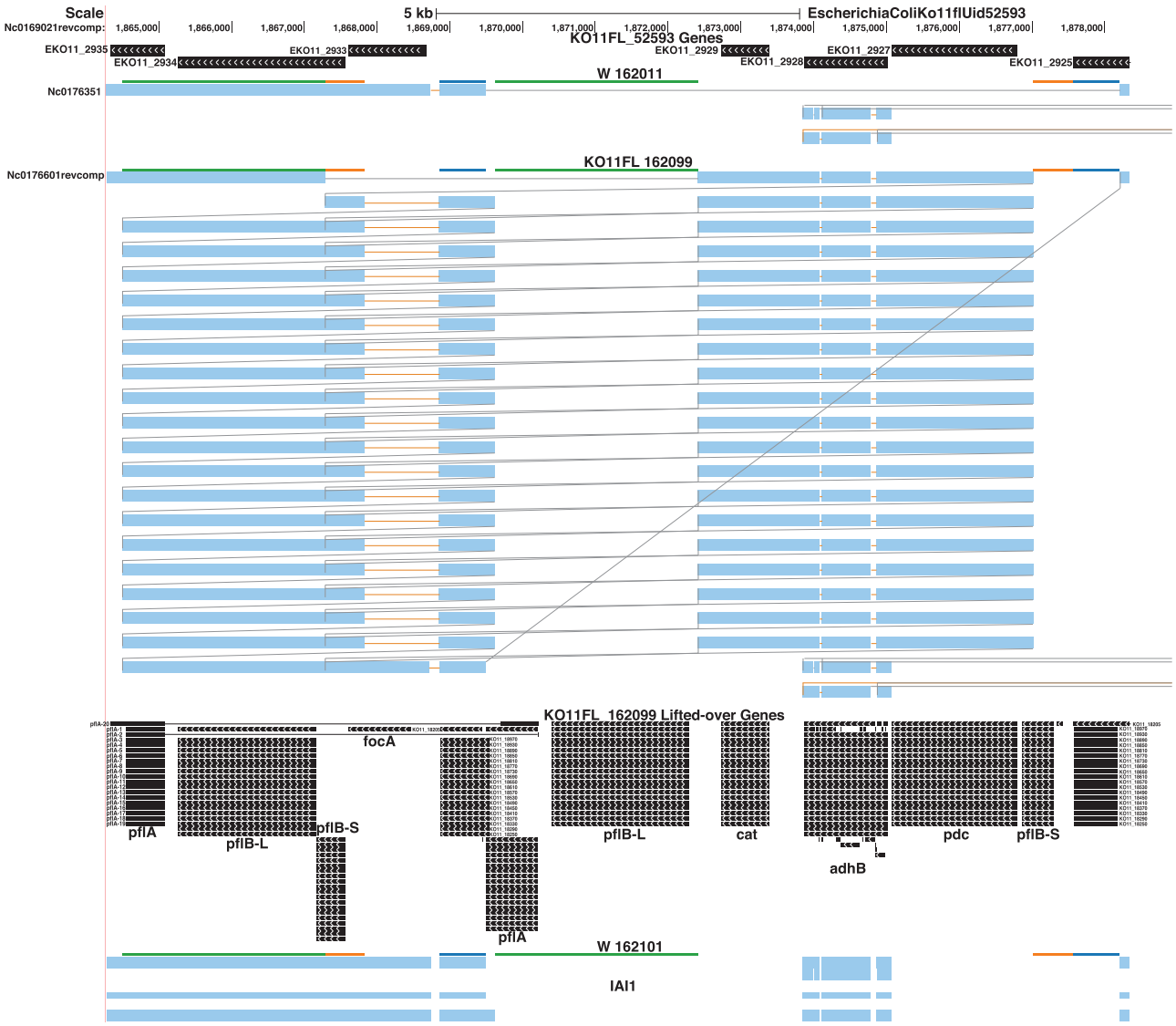


**Fig. 2.** A browser screenshot showing the *pdc-adhB-cat* tandem repeat region of *E.coli* KO11FL 162099 (Turner *et al*. 2012) displayed along the genome of *E.coli* KO11FL 52593. The colored horizontal bars on top of each snake track indicate duplications in KO11FL 52593 [two copies of each gene *pflA* (green), *pflB-L* (green) and *pflB-S* (orange)]. There is a large deletion in the parent strain W 162011, as this strain does not contain the *pdc-adhB-cat* insert. Following the snake track of KO11FL 162099, there are 20 copies of (*pflA*, *pflB-L*, *cat*, *adhB*, *pdc* and *pdfB-S*). As KO11FL 52593 has two copies of *pflA*, *pflB-L* and *pflB-S* the display arbitrarily picks one copy of each to map corresponding KO11FL 162099 orthologous genes to. The text on the screenshot was adjusted for better readability

## 2.3 Constructing the core and pan *E.coli*/*Shigella* genome

As a demonstration of the flexibility of comparative assembly hubs and the recently introduced pangenome displays it incorporates (Nguyen *et al*. 2014), we created a comparative assembly hub representing the core *E.coli*/*Shigella* genome. Briefly, this was achieved by requiring that every alignment block contains sequence from every input genome (see the Supplementary Materials). As the pangenome display algorithm generates a consensus ordering (even if the blocks correspond to the core genome rather than the wider pangenome), this is reflected in our core genome display (see Fig. 4).

We computed a core genome of size 2.7 Mb, which is consistent with the 2.7 Mb core genome for 44 *E.coli*/*Shigella* genomes reported in Sahl *et al*. (2011) and the 2.9 Mb core genome for 16 *E.coli*/*Shigella* reported in Darling *et al*. (2010). Both studies used the whole-genome alignment approach. The core genome size has been observed to decrease, but progressively plateau as the number of genomes increases [Leimbach *et al*. (2013); Lukjancenko *et al*. (2010); Touchon *et al*. (2009)]; this trend is recapitulated here, as shown in Supplementary Figure S1. In contrast to the core genome, the pangenome for the 66 strains is ~11 Mb (Supplementary Fig. S3) and continues to grow relatively linearly as new genomes are added.

The average size of an *E.coli*/*Shigella* genome is ~5 Mb, ~86% of which is genes. Approximately (assuming that the genes are evenly distributed across the genome), we expect 2.3 Mb (86% of 2.7 Mb) of the core genome to be genic, which corresponds to ~2300 genes. This is consistent with the average

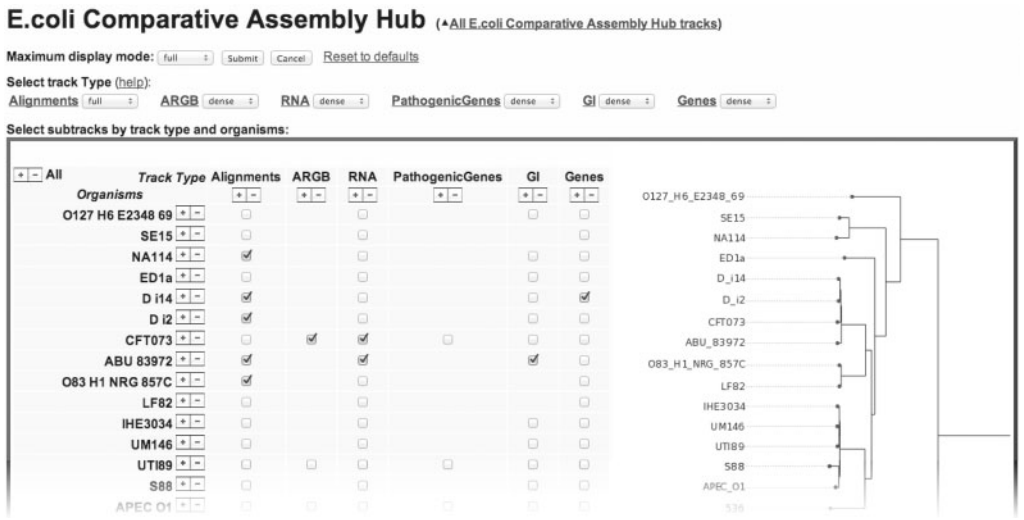


**Fig. 3.** An example portion of a comparative assembly hub configuration Web page. Each browser in the hub has its own such equivalent configuration page. Using the grid layout (rows represent the genomes, columns represent the track types), alignments and annotations can be selected regardless of which genome they were originally described on. The inset phylogenetic tree is generated automatically by the comparative assembly hub pipeline. The track controls above the grid allow quick overall configuration. Fine-grained track controls (not shown) are provided at the bottom of the page

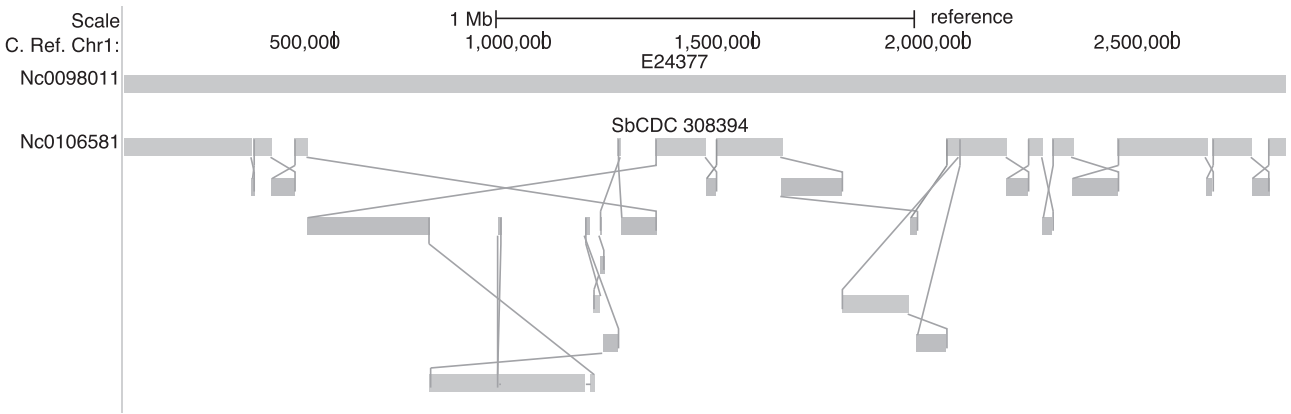


**Fig. 4.** The *E.coli*/*Shigella* core genome browser, showing the highly conserved ordering relationships between blocks of the *E.coli* core genome and the less conserved ordering in *Shigella*. Most *E.coli* look like the first snake track (E24377), with no high-level rearrangements (for space only one is shown). In contrast, the *Shigella* has, with respect to *E.coli*, a fragmented core genome (second snake track SbCDC 308394, again, only one shown for lack of space)

number of genes of each genome we observed to be overlapped with the core genome (2348 and 2507 genes for 98% and 90% minimum coverage cutoffs, respectively).

Remarkably, the core genome is, at a high level (10 kb or greater block size, approximately), entirely un-rearranged in the majority of *E.coli* genomes, despite the dramatic differences between them in their wider pangenome. The striking converse of the ordering conservation in *E.coli* is demonstrated by the *Shigella* genomes, which (as shown in the figure) are significantly reordered—though the summary allows a complete tracing of this reordering.

## 3 METHODS

Details on the presented *E.coli*/*Shigella* comparative assembly hub are in the Supplementary Materials. Below, we first give links to online installation, usage instructions and run-time analysis. We then describe the snake display algorithm, and finally describe the underlying algorithm for constructing varying levels of alignment detail for display.

### 3.1 Links to installation, usage instructions and run-time analysis

- For installing all the described software see: https://github.com/glennhickey/progressiveCactus/tree/comparative_assembly_hub_paper#installation. The git tag of progressiveCactus used is 'comparative_assembly_hub_paper'.
- For generating an HAL alignment with progressive cactus see: https://github.com/glennhickey/progressiveCactus/tree/comparative_assembly_hub_paper#running-the-aligner.
- A brief analysis of the run-time of progressive cactus for generating the described alignments and separately for aligning mammalian

genomes is at https://github.com/glennhickey/progressiveCactus/tree/comparative_assembly_hub_paper#computation-time–memory-usage.

- Basic instructions for running progressive cactus on a cluster are at: https://github.com/glennhickey/progressiveCactus/tree/comparative_assembly_hub_paper#jobtree-options-and-running-on-the-cluster.
- The manual of our cluster/compute management module, jobTree, is at: https://github.com/benedictpaten/jobTree#jobtree, the documentation for this module gives more details on supported batch systems, and links to the API for working with a new batch system.
- For alternatively generating a HAL file from a MAF file, see: https://github.com/glennhickey/hal/tree/comparative_assembly_hub_paper/README.md#maf-import.
- For building a comparative assembly hub, see: https://github.com/glennhickey/hal/tree/comparative_assembly_hub_paper/assemblyHub#comparative-assembly-hub-manual.
- For building conservation tracks (phyloP) (Hubisz *et al*. 2011) from an HAL alignment using the progressiveCactus package, see: https://github.com/glennhickey/hal/tree/comparative_assembly_hub_paper#constrained-element-prediction.

## 3.2 Snake display algorithm

A snake plot puts all the query segments within a reference chromosome range on a set of one or more horizontal rows, indexed from 1 to $n$, arranged top down. Before the algorithm is started, all the segments are placed in a list, sorted by their starting coordinate on the query, and the current row index is set to 1. Then, in a recursive fashion, the algorithm places the first segment in the list on the current row. The algorithm then iterates through the list of segments, and for each segment adds it to the current row if it does not overlap the previous segment on the current row and is either in the same order and orientation as the previous segment added to the current row, or will not have an adjacency incident with either of its sides that would connect it to a segment on the current row (thus avoiding making adjacencies between segments on a single row that are not in the same order and orientation). The current row index is then increased by 1, all segments that were placed on the previous row removed from the list, and the process repeated recursively until the list of segments is empty. Once all the segments have been drawn on their respective row, lines are drawn between the segment sides to show the adjacencies between the segments.

## 3.3 Procedural LOD algorithm

To display snake tracks usefully at all resolutions, from individual bases to whole chromosomes, we compute multiple representations of the original alignment at different LODs. Here we formalize this problem, describe our sampling-based solution and demonstrate its performance in practice.

*3.3.1 Problem definition* We begin by briefly introducing the definition of a sequence graph [more details in Paten *et al*. (2011)]. In a sequence graph $G$, each sequence $s \in S$ in the alignment (chromosome, contig, etc.) is represented by a string of DNA, which is, in turn, partitioned into segments. All homologous segments are grouped together into maximal gapless alignment blocks. Each segment within an alignment block is associated with a strand identifier to specify whether the forward or reverse strand of the segment is being aligned. Let $|G|$ refer to the number of blocks in $G$. When comparing two different sequence graphs of the same input data, $G_1$ and $G_2$, we define $\Delta(G_1, G_2)$ to be the sum of pairwise base homologies induced by the blocks of $G_1$ and not $G_2$ and those induced by the blocks of $G_2$ and not $G_1$.

Sequence graphs are conceptually equivalent to HAL graphs. Using this equivalence, we define the interpolation problem as follows. Given a sequence graph $G$ and bound $K$, compute a sequence graph $G'$ such that $|G'| \leq K$ and $\Delta(G', G)$ are minimal. In practice, $K$ is a function of the number of pixels in the browser display, and is 100 by default. Because of the size of the search space (all graphs with $\leq K$ blocks), we use the following sampling-based solution.

*3.3.2 Sampling the column graph* We use a down-sampling algorithm based on the simplifying assumption that alignment block lengths are roughly uniform: if the total length of all blocks in $G$ is $L_{tot}$, then we expect that each block in $G'$ will have length approximately $L_{block} = L_{tot}/K$. The first step of the algorithm is therefore to sample an initial graph G0 from G by sampling every $L_{block}$ bases of each $s \in S$ and extracting the block of length 1 from G into G0 (if it has not been added already).

To keep the number of sampled blocks proportional to $K$, we disregard sampled blocks whose maximum distance (along any segment) to any block already sampled is less than $L_{block}$.

*3.3.3 Extending the column graph* Unless $L_{block} = 1$, G0 will not necessarily be a valid sequence graph, as it will not contain all bases in S. We therefore greedily extend each block in G0 using the following rules, creating G1. Let $s[i]$ refer to the $i^{th}$ base of sequence s. We define a segment e as a closed interval of base coordinates $(i, j)$ where $i \leq j$ on sequence $s \in S$. Recall that because blocks are gapless, all segments contained in a given block must always have the same length. If $e$ is on the forward strand in its containing block and $s[j+1]$ exists (j + 1 is a valid coordinate) and is not already present in G1, or if $e$ is on the reverse strand and $s[i-1]$ exists and is not already in G1, then $e$ can be extended to the right. A similar check can determine whether $e$ can be extended to the left. G1 is constructed by, for each block in G0, maximally extending all segments it contains by the same length in each direction. To avoid expanding tiny gaps, we greedily extend blocks in reverse order based on the number of sequences they contain.

*3.3.4 Filling in missing blocks* We can only extend each block by the minimum length allowed for any segment it contains, and G1 will therefore still not necessarily contain every position of the input sequences. We complete the procedure by creating a block for each maximal length sequence not yet present in a block. Finally, we exhaustively merge together all pairs of blocks that have the same length such that the resulting set of pairwise homologies induced by the merged blocks are all also induced by the blocks in G.

**Algorithm 1** HAL Interpolate($G$, $K$)

```
L_tot ← sum of block lengths in G;
L_block ← L_tot/K
G' ← empty HAL graph
for s ∈ S (S is the set of sequences in G) do
  i ← 0
  while i<len(s) do
    c ← block created from alignment column in G containing s[i]
    d ← max. distance between any base in c and any base already in
    a block in G'
    if d ≤ L_block then
      G' ← G' ∪ c
    i ← i+L_block
for block b ∈ G' (from largest to smallest) do
  maximally extend b in both directions
while ∃ position x ∈ G|x∉G' do
  greedily create new block b' from x
  G' ← G' ∪ b
  maximally extend b' in both directions such that no base is present in
  two blocks of G'
```

**for** all pairs of blocks $b_1, b_2 \in G'$ **do**
  **if** all $b_1$ and $b_2$ can be merged to form a valid gapless alignment that is present in G **then**
    merge $b_1$ and $b_2$ in $G'$

*3.3.5 LOD creation* Pseudocode for the interpolation procedure described in the steps above is provided in Algorithm 1. The time complexity is $O(N+|G'|\log|G'|)$, where N is the number of bases in G and $|G'|$ is the number of blocks in the output. The O(N) component derives from the fact that each base is processed once when it is added to a block, and the $O(|G'|\log|G'|)$ component stems from the sorted sets required to extend and merge the blocks. A series of LODs can be generated from a source graph G and is created as follows. The user specifies the scaling factor between two LODs and the maximum number of blocks B to process per query, and the algorithm iteratively generates coarser LODs until one is reached such that the entire alignment can be displayed in B blocks. An API is provided such that browser queries are directed to the most detailed LOD such that the expected number of blocks returned is less than B.

*3.3.6 Experimental results* The LOD generation algorithm was used to generate the *E.coli*/*Shigella* comparative assembly hub discussed above, using default parameters that attempt to limit queries to 100 blocks per pairwise alignment. This resulted in the creation of eight LODs, whose sizes and associated query lengths are listed in Table 1. The first line of this table corresponds to the original alignment file, which is used to display browser queries of size up to 4304 bases. Each successive line corresponds to the next coarser grained LOD generated by the interpolation algorithm. We found that these LODs, whose sizes decrease exponentially, were sufficient to allow smooth transitions while zooming.

We assessed the practical impact of the LOD generation by simulating 1000 random browser queries for alignments of four *E.coli*/*Shigella* strains (W 162011, KO11FL 162099, KO11FL 52593 and Ss53) to the *E.coli* pangenome reference. The size and location of the queries were uniformly distributed across the ~10 mb of the reference genome. The hub was hosted at the San Diego Supercomputer Center (La Jolla, California, USA), whereas the Browser server was located at the University of California Santa Cruz (Santa Cruz, California, USA), and no data were cached between individual queries. Each random query was run on the hub twice: once with LOD activated (Fig. 5, blue dots) and once without (red dots). The results are grouped, by query length, into bins of size 1 000 000 along the *x*-axis where bin N contains queries in the range (N-1000000, N). The average time required for a query in each bin in seconds is reported in the figure, with the minimum and maximum query times shown in the error bars. It is apparent from the chart that without LOD, queries quickly become impractical as the length approaches a megabase, but with LOD, the time remains relatively constant for queries of any size.

**Table 1.** Size statistics for the original alignment (first row) and interpolated alignments created by the LOD algorithm

| Generated LOD | | | | |
|---|---|---|---|---|
| LOD range | $L_{block}$ | Average number of block size | Average block length | Full size |
| 1–4304 | | 109 345 | 47 | 146 |
| 4305–13 037 | 137 | 60 649 | 85 | 35 |
| 13 038–39 359 | 540 | 21 365 | 241 | 12 |
| 39 360–118 079 | 2109 | 13 071 | 394 | 7 |
| 118 080–354 485 | 8231 | 8428 | 611 | 5 |
| 354 486–1 063 580 | 32 105 | 8716 | 591 | 5 |
| 1 063 581–3 190 865 | 125 217 | 7084 | 727 | 4 |
| 3 190 866–9 572 843 | 488 359 | 4299 | 1197 | 3 |
| 9 572 844– | 1 904 608 | 2337 | 2202 | 2 |

*Notes.* Default parameters were used, setting the maximum expected blocks, *K*, to 100 and the ratio between LOD sizes to 3.9, requiring eight additional LOD alignments to be computed. *LOD range* is the query range on the browser for which the LOD applies. $L_{block}$ is the interpolation step size computed from *K*. The remaining columns report the average number of blocks per pairwise alignment, the average block length and the overall file sizes in megabytes, respectively. All block lengths are in number of bases. Because of the heuristic steps used to generate, the sampled graph (particularly 'Filling in Missing Blocks'), the number of blocks in each LOD decreases more slowly than *K*, but still exponentially.

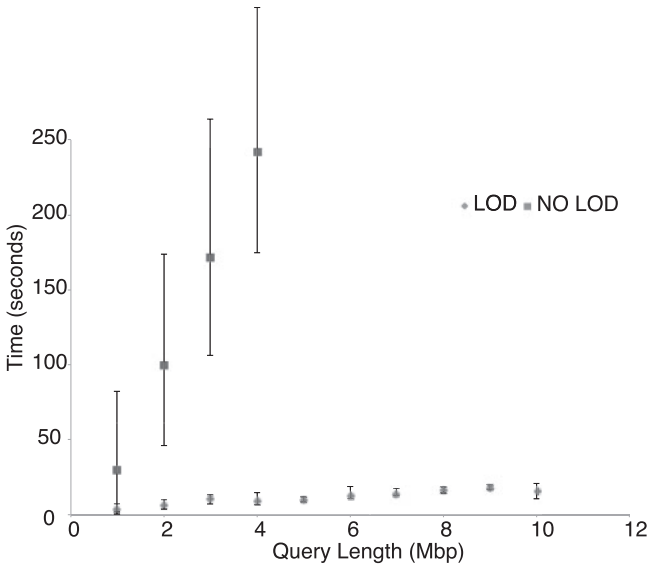


**Fig. 5.** Browser querying time with and without LODs

## 4 DISCUSSION AND CONCLUSION

In this article, we have shown how UCSC comparative assembly hubs can be easily constructed to provide useful extensible browsers for sets of evolutionarily related genomes.

The comparative assembly hub framework is novel in several respects. All the alignments and lifted-over annotations shown are mutually consistent with one another, because, for the first time, the annotation lift-over and alignment display is symmetrically driven by one reference-free alignment process, rather than a mixture of different pairwise and reference-based multiple alignments. The multiple alignment process and HAL format, being reference free, also allow us to make all the browsers equivalently powerful, in that all the annotations and alignments can be displayed from any vantage point. The snake tracks for the first time in the history of the UCSC browser and (to our knowledge) a linear display format fully express all the possible mutation types in one track, while the procedural LOD makes this useful at all resolution levels. The incorporation of the pangenome display gives a new view of the data that, for some purposes, is more useful for display than any single genome.

We used *E.coli* and *Shigella* spp. genomes as a test and automatically reconstructed a core genome of all *E.coli*/*Shigella*,

which recapitulates earlier results and visually demonstrates the many rearrangements present in the core genome of the Shigella phylogroup. Though any aligner can in principle be used, in the Supplementary Materials we demonstrate that the Cactus alignment program we used here (and which had not previously been applied to bacterial genomes) is able to successfully align the vast majority (99%) of known genes and operons in this clade.

One issue with the display of assembly hubs in general is that the data must be transferred across the Internet. By using LOD and caching we have attempted to make page load times reasonable; however, one way to avoid this bottleneck and avoid configuring the installation of a mirror Web site is to run the browser server in a virtual machine. The UCSC browser in a box scheme (http://genome.ucsc.edu/goldenPath/help/gbib), which is under development, should make this easy and be complementary to assembly hubs, allowing their display locally and potentially securely within a firewalled environment.

We have also tested comparative assembly hubs with clades of mammalian genomes (see links to online run-time analysis in the methods) therefore it is feasible to use comparative assembly hubs for even large projects, provided significant computational resources are available in the form of compute clusters. To make the tool practical for vertebrate genomics communities without these resources one future aim of the project is to provide a cloud service, where users could buy compute time to generate their alignments.

## ACKNOWLEDGEMENTS

The authors would like to thank the Howard Hughes Medical Institute. They also thank the three anonymous reviewers for providing excellent feedback.

*Funding*: The authors acknowledge the support of NIH (1U41HG007234-01) and (1U41HG006992-2) and NHGRI/NIH (5U01HG004695) for providing funding.

*Conflict of Interest*: none declared.

## REFERENCES

Darling,A.E. *et al.* (2010) progressivemauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One*, **5**, e11147.

Didelot,X. *et al.* (2012) Transforming clinical microbiology with bacterial genome sequencing. *Nat. Rev. Genet.*, **13**, 601–612.

Haas,B.J. *et al.* (2004) DAGchainer: a tool for mining segmental genome duplications and synteny. *Bioinformatics*, **20**, 3643–3646.

Hayashi,K.K. *et al.* (2006) Highly accurate genome sequences of *Escherichia coli* K-12 strains MG1655 and W3110. *Mol. Syst. Biol.*, **2,** 2006.0007.

Herbig,A. *et al.* (2012) GenomeRing: alignment visualization based on SuperGenome coordinates. *Bioinformatics*, **28**, i7–i15.

Hickey,G. *et al.* (2013) HAL: a hierarchical format for storing and analyzing multiple genome alignments. *Bioinformatics*, **29**, 1341–1342.

Hill,C.W. and Harnish,B.W. (1981) Inversions between ribosomal RNA genes of *Escherichia coli*. *Proc. Natl Acad. Sci. USA*, **78**, 7069–7072.

Hubisz,M.J. *et al.* (2011) PHAST and RPHAST: phylogenetic analysis with space/time models. *Brief. Bioinformatics*, **12**, 41–51.

Karolchik,D. *et al.* (2004) The UCSC Table Browser data retrieval tool. *Nucleic Acids Res.*, **32**, D493–D496.

Karolchik,D. *et al.* (2014) The UCSC genome browser database: 2014 update. *Nucleic Acids Res.*, **42**, D764–D770.

Krzywinski,M. *et al.* (2009) Circos: an information aesthetic for comparative genomics. *Genome Res.*, **19**, 1639–1645.

Leimbach,A. *et al.* (2013) *E. coli* as an all-rounder: the thin line between commensalism and pathogenicity. *Curr. Top. Microbiol. Immunol.*, **358**, 3–32.

Lukjancenko,O. *et al.* (2010) Comparison of 61 sequenced Escherichia coli genomes. *Microbial. Ecol.*, **60**, 708–720.

Mayor,C. *et al.* (2000) VISTA: visualizing global DNA sequence alignments of arbitrary length. *Bioinformatics*, **16**, 1046–1047.

Miller,W. *et al.* (2007) 28-way vertebrate alignment and conservation track in the UCSC genome browser. *Genes Dev.*, **17**, 1797–1808.

Nguyen,N. *et al.* (2014) Building a Pangenome Reference for a Population. In: Sharan,R. (ed.) *RECOMB*, pp. 207–221.

Nielsen,C.B. *et al.* (2010) Visualizing genomes: techniques and challenges. *Nat. Methods*, **7** (**3 Suppl.**), S5–S15.

Ogura,Y.Y. *et al.* (2009) Comparative genomics reveal the mechanism of the parallel evolution of O157 and non-O157 enterohemorrhagic. *Escherichia coli. Proc. Natl Acad. Sci. USA*, **106**, 17939–17944.

Ohta,K.K. *et al.* (1991) Genetic improvement of *Escherichia coli* for ethanol production: chromosomal integration of *Zymomonas mobilis* genes encoding pyruvate decarboxylase and alcohol dehydrogenase II. *Appl. Environ. Microbiol.*, **57**, 893–900.

Paten,B. *et al.* (2011) Cactus: algorithms for genome multiple sequence alignment. *Genome Res.*, **21**, 1512–1528.

Raney,B. *et al.* (2013) Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC genome browser. *Bioinformatics*, **30**, 1003–1005.

Rosenbloom,K.R. *et al.* (2010) ENCODE whole-genome data in the UCSC Genome Browser. *Nucleic Acids Res.*, **38** (Database issue), D620–D625.

Sahl,J.W.J. *et al.* (2011) A comparative genomic analysis of diverse clonal types of enterotoxigenic Escherichia coli reveals pathovar-specific conservation. *Infect. Immun.*, **79**, 950–960.

Thorvaldsdóttir,H.H. *et al.* (2013) Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinformatics*, **14**, 178–192.

Touchon,M. *et al.* (2009) Organised genome dynamics in the Escherichia coli species results in highly diverse adaptive paths. *PLoS Genet.*, **5**, e1000344.

Turner,P.C.P. *et al.* (2012) Optical mapping and sequencing of the *Escherichia coli* KO11 genome reveal extensive chromosomal rearrangements, and multiple tandem copies of the *Zymomonas mobilis pdc* and *adhB* genes. *J. Ind. Microbiol. Biotechnol.*, **39**, 629–639.

Waterhouse,A.M. *et al.* (2009) Jalview version 2–a multiple sequence alignment editor and analysis workbench. *Bioinformatics*, **25**, 1189–1191.

Zhu,J. *et al.* (2007) Comparative genomics search for losses of long-established genes on the human lineage. *PLoS Computat. Biol.*, **3**, e247.