hal2assemblyHub Documentation

hal2assemblyHub.py is a python wrapper that produces necessary data and files for creating the comparative assembly hubs through the UCSC genome browser. The program, which is part of the HAL tools package (`https://github.com/glennhickey/hal`), takes in the multiple sequence alignment in HAL format [1] and (optionally) any available set of annotations, either in BED or WIG format (`http://genome.ucsc.edu/FAQ/FAQformat.html`) and creates an output directory that contains all necessary data to build browsers for all input genomes and infered ancestral genomes as well as various annotation tracks. The location of the "hub.txt" file, addressable as a public URL, is pasted into the UCSC browser hub page to view the set of browsers.

Contents

1 Quick Start **3**

2 Procedural Levels of Detail **4**

3 Browsers With Annotation Tracks **5**
3.1 Annotations computed from the alignment 5
3.2 Annotations provided by users 6

4 Update Comparative Assembly Hubs **12**

5 Manipulate Hub Display **12**

6 Outputs **13**

7 Track Documentation **15**
7.1 Alignability 15
7.2 GC Percent 15
7.3 Conservation 16
7.4 Alignments and Lifted-Over Annotations 18
7.5 RepeatMasker 20

8 *hal2assemblyHub* Usage Summary **23**

1 Quick Start

From a multiple sequence alignment, users can quickly create browsers for all input genomes using the following command:

hal2assemblyHub.py <*halFile*> <*outDir*> ***–lod***

or to run in parallel:

hal2assemblyHub.py <*halFile*> <*outDir*> ***–lod –batchSystem*** <*batchType*>

or

hal2assemblyHub.py <*halFile*> <*outDir*> ***–lod –maxThreads*** <*#ofThreads*>

- *halFile* is the HAL-formatted MSA file.
- *outDir* is the output directory where all the generated files are written into. Among the output files is a file named "hub.txt", which the users will upload to the UCSC genome browser (similarly to how a track hub is created [2], see `http://genome.ucsc.edu/goldenPath/help/hgTrackHubHelp.html` for more details) and the comparative assembly hubs will be created.
- Option *–lod* is specified to compute the levels of detail, which is recommended for large datasets.
- Option *–batchSystem*: the type of batch system to run the job(s). See `https://github.com/benedictpaten/jobTree` for more details.
- Option *–maxThreads*: number of threads (processes) to use when running in single machine allows mode. Increasing this will allow more jobs to run concurrently when running on a single machine. Default=4. See `https://github.com/benedictpaten/jobTree` for more details.

Examples:

hal2assemblyHub.py *alignment.hal outdir* ***–lod***
hal2assemblyHub.py *alignment.hal outdir* ***–lod –batchSystem*** *gridEngine*
hal2assemblyHub.py *alignment.hal outdir* ***–lod –maxThreads*** *24*

2 Procedural Levels of Detail

To improve browsing speed, especially for browsing at all levels of resolution (from individual bases to whole chromosomes) of large datasets, we compute multiple representations of the original alignment at different levels of detail. HAL Tools provide programs to resample a HAL graph to compute coarser-grained levels of detail to speed up subsequent analysis at different scales. Please see `https://github.com/glennhickey/hal/`, "Levels of Detail" for more information.

hal2assemblyHub takes care of generating levels of detail for the alignment if option *–lod* is specified. Example:

> ***hal2assemblyHub.py*** *alignment.hal outdir* ***–lod***

By default, no level of detail is generated. Users can independently generate them using the HAL Tools' *halLodInterpolate.py*. Generating levels of detail can be time-consuming. Users can provide *hal2assemblyHub* with pre-computed levels of detail (and avoid re-computing them) by the options *–lodTxtFile* and *–lodDir*:

> ***hal2assemblyHub.py*** *alignment.hal outdir* ***–lod*** ***–lodTxtFile*** *lod.txt* ***–lodDir*** *loddir/*

lod.txt and *loddir* are output files of *halLodInterpolate.py*. See "*halLodInterpolate.py -h*" for explanations of more options.

- *lod.txt* is output text file with links to interpolated hal files, with each file is associated a value stating its minimum suggested query range (in bases).
- *loddir* is the path of the directory where interpolated hal files are stored.

3 Browsers With Annotation Tracks

3.1 Annotations computed from the alignment

hal2assemblyHub.py takes care of computing various annotation tracks from the alignment, including "Alignability", "Conservation", "GC Content" and "Clade Exclusive Regions".

Examples:

hal2assemblyHub.py *alignment.hal outdir* ***–lod –cladeExclusiveRegions –alignability –gcContent –conservation*** *conservationRegions.bed* ***–conservationGenomeName*** *hg19*

- *–cladeExclusiveRegions*: for each node in the phylogenetic tree of the genomes in the alignment, regions that are genome-specific (leaf-node) or clade-specific (internal node), i.e. present only in genomes within the clade and absent in other genomes, are printed out in bigbed-formatted files. These files will be located at $outdir/liftoverbed/CladeExclusive$. The resulting comparative assembly hub contains one track for each genome. See *–maxOutgroupGenomes* and *–minIngroupGenomes* options (in the Usage Summary section 8 below) for adjusting the definition of "clade exclusive".

- *–alignability*: for each node GENOME in the tree (each genome and each ancestral genome), computes the wiggle track (the Alignability track) of the number of other unique genomes (including ancestral genomes) each base aligns to. The computed bigwig-formatted file is located at $outdir/GENOME/GENOME.alignability.bw$.

- *–gcContent*: for each node GENOME, computes the GC content (gcContent track) of that genome, following instructions at `http://genomewiki.ucsc.edu/index.php/Browser_Track_Construction#GC_Percent`. The computed bigwig file is located at $outdir/GENOME/GENOME.gc.bw$.

- *–conservation*: computes the conservation track for each genome, showing measurements of evolutionary conservation using the phyloP program from the PHAST package. See Section 7 for more details. *conservationRegions.bed* is the bed-formatted file (see `http://genome.ucsc.edu/FAQ/FAQformat.html#format1`) providing neutral regions of a reference genome for creating a neutral model. By default, it expects the neutral regions to be coding genes and uses 4fold degenerate site within those genes (which it extracts automatically). The neutral regions could also be ancestral repeats (or anything else). An example of this file:

  chr1 363 2826 gene1
  chr1 2827 3760 gene2

The fields above are $< Chromosome >$, $< Start-coordinate >$, $< End-coordinate >$ and $< Gene - name >$. The chromosome and the coordinates must be consistent with the input HAL file (e.g. same chromosome name, an example of a common inconsistency is when the HAL file has "1" and the bed file has "chr1").

- *–conservationGenomeName*: name of the reference genome used to provide the neutral region information in $--conservation$. This must be consistent with the genome name in HAL file as well.
  The computed conservation tracks are stored at *outdir/conservation*.

Computing conservation scores could be expensive. Use option *–conservationDir* to use pre-computed conservation scores:

hal2assemblyHub.py *alignment.hal outdir* ***–lod –conservationDir*** *myConservationDir/*

- *–conservationDir*: Directory contains conservation bigwigs. Format:
  myConservationDir/

  Genome1_phyloP.bw

  Genome2_phyloP.bw

  ...

  *Genome*1, *Genome*2, etc. are genome names and must be consistent with names in the input HAL file.

3.2 Annotations provided by users

Currently, *hal2assemblyHub.py* supports two annotation formats: *bed* (or big bed) and *wiggle* (or bigwig) (see `http://genome.ucsc.edu/FAQ/FAQformat.html`). Example annotations are genes, transcription levels, histone modifications, etc.

Example 1:

hal2assemblyHub.py *alignment.hal outdir* ***–lod –bedDirs*** *Genes* ***–tabBed***

- *–bedDirs*: comma separated list of paths to different annotation directories, one directory per annotation type. In this example, there is only one annotation type, which is *Genes*. The format of each annotation directory is:

  Genes/
  - Genome1/
    - bedfile1.bed
    - bedfile2.bed
    - ...
  - Genome2/
    - ...
  - ...

  *bedfile*1.*bed* and *bedfile*2.*bed* are gene annotations of *Genome*1. By default, these annotations will be lifted-over(/mapped/translated) to all other genomes in the alignment, unless options *–noBedLiftover* is specified. Please only include genomes that have annotations. For example, if only *Genome*1 and *Genome*2 have gene annotations, the *Genes/* directory should only have *Genome*1/ and *Genome*2/. Note, names of *Genome*1 and *Genome*2 must be consistent with the genome names in the alignment. The bed file names may be anything as long as they have they *.bed* extension. The name of each annotation directory (*Genes* in this case) will be used as the track name on the browser. For example, browser of *Genome*1 will have a track named "*Genome1 Genes*" and a track named "*Genome2 Lifted-over Genes*".

- –tabBed: if the input bed files are tab-separated (recommended), this option must be specified. The default settings assume space-delimited. If the bed files are space-delimited, the field values must not contain any space.

Example 2:

hal2assemblyHub.py *alignment.hal outdir* ***–lod –bedDirs*** *allAnnotations/Genes,allAnnotations/CpG-Islands,allAnnotations/Variations* ***–tabBed***

In this example, there are three different annotation types: *Genes*, *CpG-Islands*, and *Variations*, all located within the directory *allAnnotations/*.

Example 3:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–bedDirs*** *Genes,CpG-Islands,Variations* ***–tabBed*** ***–noBedLiftover***

- *–noBedLiftover*: if specified, the $lift-over$ step is disable, i.e. only creates track for the input annotations and does not lift/map these annotations to other genomes.

Example 4:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–finalBigBedDirs*** *Genes,CpG-Islands,Variations* ***–tabBed***

- *–finalBigBedDirs*: comma separated list of directories containing final big bed files to be displayed. No liftover will be done for these files. Each directory represents a type of annotation. This option is useful when annotations have been previous lifted-over and can just be fed to the pipeline, to avoid rerunning the lift-over processes. Format of each directory:
  bbDir/
  queryGenome1/
  targetGenome1.bb
  targetGenome2.bb
  ...
  queryGenome2/
  ...
  Annotations of queryGenome have been lifted-over (mapped) to targetGenomes and will be displayed on each targetGenome's browser. For example, if *bbDir* is *Genes*, *targetGenome*1.*bb* contains the gene annotations of *queryGenome*1 mapped to *targetGenome*1, in bigBed format. *queryGenome* and *targetGenome* are the same for the original (non lifted-over) annotations (e.g. gene annotations of *queryGenome*1).

  Note: it is **not** required that each annotation must be lifted over to all other genomes. The pipeline prepares one track for each bigBed file - users can choose which tracks to include.

Example 5:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–bedDirs*** *Genes* ***–finalBigBedDirs*** *CpG-Islands,Variations* ***–tabBed***

In this example, the pipeline will not perform lifting-over for the *CpG-Islands* and *Variations* annotations (in bigBed format) - the corresponding tracks will be displayed on the resulting comparative hubs "as is", while the *Genes* annotations (in bed format) will be lifted-over. This is applicable when users wish to include new annotations into their comparative assembly hubs, or to update some annotations while keeping the rest intact.

Example 6:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–bedDirs*** *Genes,CpG-Islands* ***–bedDirs2*** *Variations* ***–tabBed***

- *–bedDirs2*: Similar to *–bedDirs*, except the tracks for the annotations specified here will be kept separately and out of the composite track. In this case, the *Genes* and *CpG-Islands* tracks will be included in the composite track (hubCentral) while the *Variations* tracks will be on its own.

Example 7:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–finalBigBedDirs*** *Genes,CpG-Islands* ***–finalBigBedDirs2*** *Variations* ***–tabBed***

- *–finalBigBedDirs2*: Similar to *–finalBigBedDirs*, except these tracks will be kept separately and out of the composite track.

Example 8:

hal2assemblyHub.py *alignment.hal outdir* ***–lod*** ***–bedDirs*** *Genes,CpG-Islands* ***–tabBed*** ***–wigDirs*** *Transcription,Methylation*

- *–wigDirs*: similar to *–bedDirs*, but for wiggle format files.

Item Searching of Annotation Tracks

By default, *hal2assemblyHub.py* index the *name* column of the input bed files so that when browsing the hubs, users can quickly search for specific items using their *name*s. Additional fields can be added to the bed files and the pipeline will index them for searching. When there are additional fields in the bed files, an "*.as*" (AutoSQL) format file is required for each input bed directory. See `http://genome.ucsc.edu/goldenPath/help/bigBed.html#Ex3` for the format of the ".as" file.

This is applicable when users want to be able to search genes by various IDs, such as accession numbers and common names. If the *name* column in the bed file is the accession number, add an additional field *common-name* to the bed file, and use the *.as* file to specify this additional field. In **Example1**, the input *Genes* directory will be as followed:

- Genes/

  Genome1/
  myAsFile.as
  bedfile1.bed
  bedfile2.bed
  ...
  Genome2/
  anotherAsFile.as
  ...
  ...

Example of an *.as* file:

```
table geneEscherichiaColi042Uid161985
"EscherichiaColi042Uid161985 genes with additional fields commonName, synonym
and product"
(
string chrom; "Reference sequence chromosome or scaffold"
uint chromStart; "Start position of feature on chromosome"
uint chromEnd; "End position of feature on chromosome"
string name; "Name of gene"
uint score; "Score"
char[1] strand; "+ or - for strand"
uint thickStart; "Coding region start"
uint thickEnd; "Coding region end"
uint reserved; "RGB value"
int blockCount; "Number of blocks"
int[blockCount] blockSizes; "A comma-separated list of block sizes"
```

```
int[blockCount] chromStarts; "A comma-separated list of block starts"
string commonName; "Gene common name"
string synonym; "Gene synonym"
string product; "Gene product"
)
```

In this case, there are three extra fields: *commonName*, *synonym* and *product*, and those three fields, together with the *name* field, will be indexed for searching, i.e. when browsing the resulting hub browsers, users can search a gene by its name, common name, synonym or product.

4 Update Comparative Assembly Hubs

The simplest way to update comparative assembly hubs is to rerun *hal2assemblyHub.py* and utilize the following options:

- *–twobitdir*
- *–lodTxtFile*
- *–lodDir*
- *–finalBigBedDirs*
- –finalBigBedDirs2
- *–conservationDir*

See the above sections and the Usage Summary section 8 for more details.

5 Manipulate Hub Display

To manipulate the hub displays, see the following options:

- *–hub*
- *–shortLabel*
- *–longLabel*
- *–email*
- *–genomes*
- *–rename*
- *–tree*
- *–url*

6 Outputs

Comparative Assembly Hubs are built utilizing the Assembly Hub function of the UCSC Genome Browser. Many of the output files produced by the Comparative Assembly Hub Pipeline are explained in details here: `http://genomewiki.ucsc.edu/index.php/Assembly_Hubs`. To avoid potential problems, we recommend users to provide an empty *outdir* when running *hal2assemblyHub.py*.

The output directory may contain:

1. *hub.txt*: The primary URL reference for the constructed comparative assembly hubs. Please paste the URL of the location of this file to the UCSC genome browser to load the hubs. This is similar to how a track hub is created, please see `http://genome.ucsc.edu/goldenPath/help/hgTrackHubHelp.html` for more instructions. This file contains a short description of the hub properties, including the hub name, short label, long label and contact email.
2. *genomes.txt*: list of genome assemblies included in the hub.
3. *groups.txt*: definitions of track groups. Track groups are the sections of related tracks grouped together under the primary genome browser graphics display image.
4. Genome assembly directories: one directory is created for each genome assembly, one directory for each ancestral genome, and one for the pangenome, if appropriate. Example:

*GenomeAssembly*1/

*GenomeAssembly*1.2*bit*

chrom.sizes

trackDb.txt

description.html

*GenomeAssembly*1.*alignability.bw*: Bigwig file for the alignability track of GenomeAssembly1, generated if option *–alignability* is specified when running hal2assemblyHub.py. Alignability is the number of genome assemblies that have bases aligned with each base of the current assembly (mappability).

*GenomeAssembly*1.*gc.bw*: Bigwig file for the GC Content track of GenomeAssembly1, generated if options *–gcContent* is specified when running hal2assemblyHub.py.

repeatMasker/: subdirectory containing repeatMasker files of the GenomeAssembly1, present if option *–rmskDir* is specified when running *hal2asssemblyHub.py*.

*** For more details on items (1) to (4), see: http://genome.ucsc.edu/goldenPath/help/hgTrackHubHelp.htmlSetup ***

5. *conservation/*: Files necessary for the Conservation Track of each GenomeAssembly Browser, generated if option –conservation is used.

6. *hubTree.png*: Phylogenetic tree image of the genome assemblies that is displayed in the configuration page of each genome assembly's hub browser.

7. *liftoverbed/*: All bed annotation files, including both input bed files and lifted-over bed files. Example:

*Annotation*1/

*GenomeAssembly*1/

*GenomeAssembly*1.*bb* : annotation1 of GenomeAssembly1

*GenomeAssembly*2.*bb* : annotation1 of GenomeAssembly2 mapped onto GenomeAssembly1

...

*GenomeAssembly*2/

...

...

Annotation#/

8. *documentation/*: documentation files automatically generated by *hal2assemblyHub.py*. These files are used for documentation of the various tracks on the hub browsers (see Section 7).

9. *lod.txt*: (Level of details) the lod text file generated by *halLodInterpolate.py*, or by the pipeline (which calls *halLodInterpolate.py*) if option *–lod* is specified. The text file contains links to interpolated hal files, with each file is associated a value stating its minimum suggested query range (in bases).

10. *lod/*: the output directory of *halLodInterpolate.py*, containing the interpolated lod files

11. *alignment.hal*: the multiple sequence alignment of the input genome assemblies, in HAL format.

7 Track Documentation

The following track documentation are automatically generated by *hal2assemblyHub.py* with each corresponding track, and is be displayed by the browser on the track information page.

7.1 Alignability

The documentation for the Alignability track of all genomes is located at *outdir/documentation/alignability.html*. To edit the track documentation, please edit the *alignability.html* file.

Description

This track shows the number of genomes aligned to each position of the reference. The values range from 0 to the total number of input genomes and imputed ancestral genomes.

Methods

Alignability was generated using the halAlignability script of the HAL tools package.

References

Hickey *etal*. HAL: a hierarchical format for storing and analyzing multiple genome alignments. *Bioinformatics*. 2013 May;29(10):1341-1342

7.2 GC Percent

The documentation for the gcPercent track of all genomes is located at *outdir/documentation/gcPercent.html*. To edit the track documentation, please edit the *gcPercent.html* file.

Description

The GC percent track shows the percentage of G (guanine) and C (cytosine) bases in 5-base windows. High GC content is typically associated with gene-rich areas.

This track may be configured in a variety of ways to highlight different aspects of the displayed information. Click the "Graph configuration help" link for an explanation of the configuration options.

Methods

This track was generated following the UCSC GC_Percent Track Construction instructions (`http://genomewiki.ucsc.edu/index.php/Browser_Track_Construction#GC_Percent`), using the sequence information extracted from the multiple sequence alignments.

References

The GC Percent graph presentation is by Hiram Clawson. The data was automatically generated using the HAL tools package.

7.3 Conservation

The documentation for the Conservation track of all genomes is located at *outdir/documentation/conservation.html*. To edit the track documentation, please edit the *conservation.html* file.

Description

This track shows measurements of evolutionary conservation using the *phyloP* program from the PHAST package (`http://compgen.bscb.cornell.edu/phast/`), for all genomes in the comparative assembly hub. The multiple alignments were generated using progressiveCactus.

PhyloP separately measures conservation at individual columns, ignoring the effects of their neighbors. PhyloP is appropriate for evaluating signatures of selection at particular nucleotides or classes of nucleotides (e.g., third codon positions, or first positions of miRNA target sites). PhyloP can measure acceleration (faster evolution than expected under neutral drift) as well as conservation (slower than expected evolution). In the phyloP plots, sites predicted to be conserved are assigned positive scores (and shown in blue), while sites predicted to be fast-evolving are assigned negative scores (and shown in red). The absolute values of the scores represent -log p-values under a null hypothesis of neutral evolution. PhyloP treat alignment gaps and unaligned nucleotides as missing data.

Display Convention and Configuration

In full and pack display modes, conservation scores are displayed as a *wiggletrack* (histogram) in which the height reflects the size of the score. The conservation wiggles can be configured in a variety of ways to highlight different aspects of the displayed information. Click the Graph configuration help link (`http://genome.ucsc.edu/goldenPath/help/hgWiggleTrackHelp.html`) for an explanation of the configuration options.

Methods

The conservation tracks of this comparative assembly hub were created using the phyloP package (`https://github.com/glennhickey/hal/tree/development/phyloP`), which is part of HAL tools. The HAL's phyloP is a python wrapper for running the PHAST package phyloP program and building conservation tracks for all genomes in the HAL multiple alignment. The process starts with creating a neutral model using a reference genome and neutral regions provided. By default, it expects the neutral regions to be coding genes and uses 4fold degenerate site within those genes (which it extracts automatically). The neutral regions could also be ancestral repeats (or anything else). After the model is created, the program proceeds to compute the conservation scores for positions along the root genome. These scores are subsequently lifted over to the children genomes using the multiple alignment. Lastly, HAL phyloP computes the conservation scores for regions in the children genomes that do not align to the root genome. (Of note, the program uses the phylogenetic tree extracted from the HAL file if the tree is not specified.)

Credits

We thank Melissa Jane Hubisz and Adam Siepel for the PHAST package and their help with HAL phyloP.
The HAL phyloP package: Glenn Hickey, Joel Armstrong, Ngan Nguyen, Benedict Paten.

References

Siepel, A., Pollard, K. and Haussler, D.. New methods for detecting lineage-specific selection. *ResearchinComputationalMolecularBiology*. 2006:190-205.

Hickey *etal*.. HAL: a hierarchical format for storing and analyzing multiple genome alignments. *Bioinformatics*. 2013 May;29(10):1341-1342.

7.4 Alignments and Lifted-Over Annotations

The documentation for the Alignment *snake* tracks, lifted-over annotation tracks and all other tracks in the hubCentral is located at *outdir/documentation/hubCentral.html*. To edit the documentation, please edit the *hubCentral.html* file.

Alignments

Description

An alignment track, or *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.

Display Convention and Configuration

In *full* display mode, 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 target genome aligned to the given portion of the reference genome. Adjacencies represent the covalent bonds between the aligned subsequences of the target genome. Segments can be configured to be colored by chromosome, strand or left a single color under the *SelecttrackType*, *Alignments*, then *Blockcoloringmethod*.

Red tick-marks within segments represent substitutions with respect to the reference, shown in windows of the reference of (by default) up to 50 kilobases. This default can be adjusted under *SelecttrackType*, *Alignments*, then *Maximumwindowsizeinwhichtoshowmismatches*. Zoomed in to the base-level these substitutions are labeled with the non-reference base.

An insertion in the reference relative to the target creates a gap between abutting segment sides that is connected by an adjacency. An insertion in the target relative to the reference is represented by an orange tick mark that splits a segment at the location the extra bases would be inserted. Simultaneous independent insertions in both target and reference look like an insertion in the reference relative to the target, except that the corresponding adjacency connecting the two segments is colored orange. More complex structural rearrangements create adjacencies that connect the sides of non-abutting segments in a natural fashion.

Duplications within the target genome create extra segments that overlap along the reference genome axis. Duplications within the reference imply self-alignments, intervals of the reference genome that align to other intervals of the

reference genome. To show these self-alignments within the reference genome we draw colored coded sets of lines along the reference genome axis that indicate these self homologies, and align any target segments that align to these regions arbitrarily to just one copy of the reference self alignment.

The *pack* display option can be used to display a larger number of Snake tracks in limited vertical browser. This mode 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 target sequence.

The *dense* display further eliminates these duplications so that each Snake track is compactly represented along just one row.

To ensure that the snake alignments track loads quickly at any resolution, from windows showing individual bases up to entire scaffolds or chromosomes, the LOD (Levels-Of-Detail) algorithm (part of the HAL tools package) is used, which creates scaleable levels of detail for the alignments. The additional use of the hdf5 caching scheme further aides scaling.

Various mouse overs are implemented and clicking on segments navigates to the corresponding region in the target genome, making it simple to instantly switch the alignment view between reference points.

Methods

A *snake* is a way of viewing a set of pairwise gap-less alignments that may overlap on both the reference and query genomes. Alignments are always represented as being on the positive strand of the reference species, but can be on either strand on the query sequence.

A *snake* plot puts all the query segments within a reference chromosome range on a set of one or more levels. All the segments on a level are on the same strand, do not overlap in reference coordinate space, and are in the same order and orientation in both sequences. This is the same requirement as the alignments in a chain on the UCSC browser. Before the algorithm is started, all the segments are sorted by their starting coordinate on the query, and the current level is set to one. Then in a recursive fashion, the algorithm places the first segment on the current list on the current level, and then adds all the rest of the segments on the list that will fit onto the current level with the requirements that all the segments on a level are on the same strand, and that the proposed segment be non-overlapping and have a reference start address that is greater than the query end address of the previously added segment on that level. All segments that will not fit on the current level are then added to subsequent levels following the same rules. Once all the segments have been assigned a level, lines are drawn between the segments to

show the adjacencies in the list when sorted by query start address.

Credits

The *snake* alignment display was implemented by Brian Raney.
HAL supports and track generations: Glenn Hickey, Ngan Nguyen, Joel Armstrong, Benedict Paten.

Lifted-over Annotations

Description

Lifted-over annotation tracks show the annotations of any genome translated onto the reference genome, via a process of lift-over. All the alignments and lifted over annotations shown are mutually consistent with one another, because 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.

Methods

The lifted-over tracks were generated using the $halLiftover$ and/or the $halWiggleLiftover$ scripts of the HAL tools package.

Credits

Glenn Hickey, Ngan Nguyen, Joel Armstrong, Benedict Paten.

References

Hickey *etal*.. HAL: a hierarchical format for storing and analyzing multiple genome alignments. $Bioinformatics$. 2013 May;29(10):1341-1342.

Comparative Assembly Hubs: Web Accessible Browsers for Comparative Genomics

7.5 RepeatMasker

The documentation for the repeatMasker track of all genomes is located at $outdir/documentation/repeatMasker.html$. To edit the documentation, please edit the $repeatMasker.html$ file.

Description

This track was created by using Arian Smit's RepeatMasker program (`http://www.repeatmasker.org/`), which screens DNA sequences for interspersed repeats and low complexity DNA sequences. The program outputs a detailed annotation of the repeats that are present in the query sequence (represented by this track), as well as a modified version of the query sequence in which all the annotated repeats have been masked. RepeatMasker uses the Repbase Update (`http://www.girinst.org/repbase/update/index.html`) library of repeats from the Genetic Information Research Institute (GIRI). Repbase Update is described in Jurka (2000) in the References section below.

Display Conventions and Configuration

In full display mode, this track displays up to ten different classes of repeats:

Short interspersed nuclear elements (SINE), which include ALUs
Long interspersed nuclear elements (LINE)
Long terminal repeat elements (LTR), which include retroposons
DNA repeat elements (DNA)
Simple repeats (micro-satellites)
Low complexity repeats
Satellite repeats
RNA repeats (including RNA, tRNA, rRNA, snRNA, scRNA, srpRNA)
Other repeats, which includes class RC (Rolling Circle)
Unknown

The level of color shading in the graphical display reflects the amount of base mismatch, base deletion, and base insertion associated with a repeat element. The higher the combined number of these, the lighter the shading.

Methods

Data are generated using the RepeatMasker. Repeats are soft-masked. Alignments may extend through repeats, but are not permitted to initiate in them.

Credits

Thanks to Arian Smit, Robert Hubley and GIRI for providing the tools and repeat libraries used to generate this track.

References

Smit AFA, Hubley R, Green P. RepeatMasker Open-3.0. `http://www.repeatmasker.org`. 1996-2010.

Repbase Update is described in:

Jurka J. Repbase Update: a database and an electronic journal of repetitive elements. *TrendsGenet*. 2000 Sep;16(9):418-420. PMID: 10973072

For a discussion of repeats in mammalian genomes, see:

Smit AF. Interspersed repeats and other mementos of transposable elements in mammalian genomes. *CurrOpinGenetDev*. 1999 Dec;9(6):657-63. PMID: 10607616

Smit AF. The origin of interspersed repeats in the human genome. *CurrOpinGenetDev*. 1996 Dec;6(6):743-8. PMID: 8994846

8 *hal2assemblyHub* Usage Summary

Usage: hal2assemblyHub.py <halFile> <outputDirectory> [options]

Options:	
-h, –help	show this help message and exit
–cpHalFileToOut	If specified, copy the input halfile to the output directory (instead of just make a softlink). Default=False
HUB INFORMATION:	
–hub=HUBLABEL	a single-word name of the directory containing the track hub files. Not displayed to hub users. Default=myHub
–shortLabel=SHORTLABEL	the short name for the track hub. Suggested maximum length is 17 characters. Displayed as the hub name on the Track Hubs page and the track group name on the browser tracks page. Default=my hub
–longLabel=LONGLABEL	a longer descriptive label for the track hub. Suggested maximum length is 80 characters. Displayed in the description field on the Track Hubs page. Default=my hub
–email=EMAIL	the contact to whom questions regarding the track hub should be directed. Default=NoEmail
–genomes=GENOMES	File specified list of genomes to make browser for. If specified, only create browsers for these genomes in the order provided by the list. Otherwise create browsers for all genomes in the input hal file
–rename=RENAME	File that maps halfile genomeNames to names displayed on the browser. Format: <halGenomeName>tab<genomeNameToDisplayOnBrowser>. Default=none
–tree=TREEFILE	Newick binary tree. The order of the tracks and the default track layout will be based on this tree if option “genomes” is not specified. If not specified, try to extract the newick tree from the input halfile.
–url=URL	Public url of the hub location
–twobitdir=TWOBITDIR	Optional. Directory containing the 2bit files of each genomes. Default: extract from the input hal file.

LEVEL OF DETAILS:	
Level-of-detail (LOD) options	
–lod	If specified, create "level of detail" (lod) hal files and will put the lod.txt at the bigUrl instead of the original hal file. Default=False
–lodTxtFile=LODTXTFILE	"hal Level of detail" lod text file. If specified, will put this at the bigUrl instead of the hal file. Default=none
–lodDir=LODDIR	"hal Level of detail" lod dir. If specified, will put this at the bigUrl instead of the hal file. Default=none
–lodMaxBlock=LODMAXBLOCK	Maximum number of blocks to display in a hal level of detail. Default=none
–lodScale=LODSCALE	Scaling factor between two successive levels of detail. Default=none.
–lodMaxDNA=LODMAXDNA	Maximum query length that will such that its hal level of detail will contain nucleotide information. Default=none.
–lodInMemory	Load entire hal file into memory when generating levels of detail instead of using hdf5 cache. Default=False.
–lodNumProc=LODNUMPROC	Number of levels of detail to generate concurrently in parallel processes
–lodMinSeqFrac=LODMINSEQFRAC	Minumum sequence length to sample as fraction of step size for level of detail generation: ie sequences with length $\leq$ floor(minSeqFrac $*$ step) are ignored. Use default from halLodExtract if not set.
–lodChunk=LODCHUNK	HDF5 chunk size for generated levels of detail.

BED-FORMATTED ANNOTATIONS:

All annotations in bed or bigbed formats.

–bedDirs=BEDDIRS	comma separated list of directories containing bed files of the input genomes. Each directory represents a type of annotation. The annotations of each genome will then be liftovered to all other genomes in the MSA. Example: "genes,genomicIsland,tRNA". Format of each directory: bedDir/ then genome1/ then chr1.bed, chr2.bed... Default=none
–finalBigBedDirs=BBDIRS	comma separated list of directories containing final big bed files to be displayed. No liftover will be done for these files. Each directory represents a type of annotation. Example: "genes,genomicIsland,tRNA". Format of each directory: bbDir/ then queryGenome/ then targetGenome1.bb, targetGenome2.bb ... (so annotation of queryGenome has been mapped to targetGenomes and will be display on the targetGenome browsers). Default=none
–bedDirs2=BEDDIRS2	Similar to –bedDirs, except these tracks will be kept separately and out of the composite track. Default=none.
–finalBigBedDirs2=BBDIRS2	Similar to –finalBigBedDirs, except these tracks will be kept separately and out of the composite track. Default=none.
–noBedLiftover	If specified, will not lift over the bed annotations. Default=False.
–tabBed	If specified, treat tab as the delimiter of all the bed files. Default: any white space.

WIGGLE-FORMATTED ANNOTATIONS:

All annotations in wiggle or bigWig formats.

–wigDirs=WIGDIRS	comma separated list of directories containing wig files of the input genomes. Each directory represents a type of annotation. The annotations of each genome will then be liftovered to all other genomes in the MSA. Example: "genes,genomicIsland,tRNA". Format of each directory: wigDir/ then genome1/ then chr1.wig, chr2.wig... Default=none
–finalBigwigDirs=BWDIRS	comma separated list of directories containing final big wig files to be displayed. No liftover will be done for these files. Each directory represents a type of annotation. Example: "readCoverage,". Format of each directory: bwDir/ then queryGenome/ then targetGenome1.bw, targetGenome2.bw ... (so annotation of queryGenome has been mapped to targetGenomes and will be display on the targetGenome browsers). Default=none.
–nowigLiftover	If specified, will not lift over the wig annotations. Default=False

REPEATMASKER:

–rmskDir=RMSKDIR Directory containing repeatMasker's output files for each genome. Format: rmskDir/ then genome1/ then genome.rmsk.SINE.bb, genome.rmsk.LINE.bb, ... Default=none

GC PERCENT:

–gcContent If specified, make GC-content tracks. Default=False

ALIGNABILITY:

–alignability If specified, make Alignability tracks. Default=False

CONSERVATION TRACKS:

Necessary information for computing conservation tracks

–conservation=CONSERVATION
Bed file providing regions to calculate the conservation tracks.

–conservationDir=CONSERVATIONDIR
Optional. Directory contains conservation bigwigs if available. These bigwigs will be used. If this is not specified, the program will compute the conservation tracks.

–conservationGenomeName=CONSERVATIONGENOMENAME
Name of the genome of the bed file provided in the "–conversation" option.

–conservationTree=CONSERVATIONTREE
Optional. Newick tree for the conservation track.

–conservationNumProc=CONSERVATIONNUMPROC
Optional. Number of processors to run conservation

CLADE EXCLUSIVE REGIONS:

Requirements of regions that are exclusive to subgroup of genomes.

–cladeExclusiveRegions If specified, will generate tracks of regions that are exclusive to each branch (including leaf "branches", which will be genome-exclusive regions) on the tree. Default=False

–maxOutgroupGenomes=MAXOUT
Maximum number of outgroup genomes that a region is allowed to be in. Default=0

–minIngroupGenomes=MININ
Minimum number of ingroup genomes that a region must appear in. Default=all ingroup genomes (branch node and all its children).

References

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

[2] Raney, B.J., Dreszer, T.R., Barber, G.P., Clawson, H., Fujita, P.A., Wang, T., Nguyen, N., Paten, B., Zweig, A.S., Karolchik, D., et al.: Track data hubs enable visualization of user-defined genome-wide annotations on the ucsc genome browser. Bioinformatics **30**(7) (2014) 1003–1005