harvest Documentation

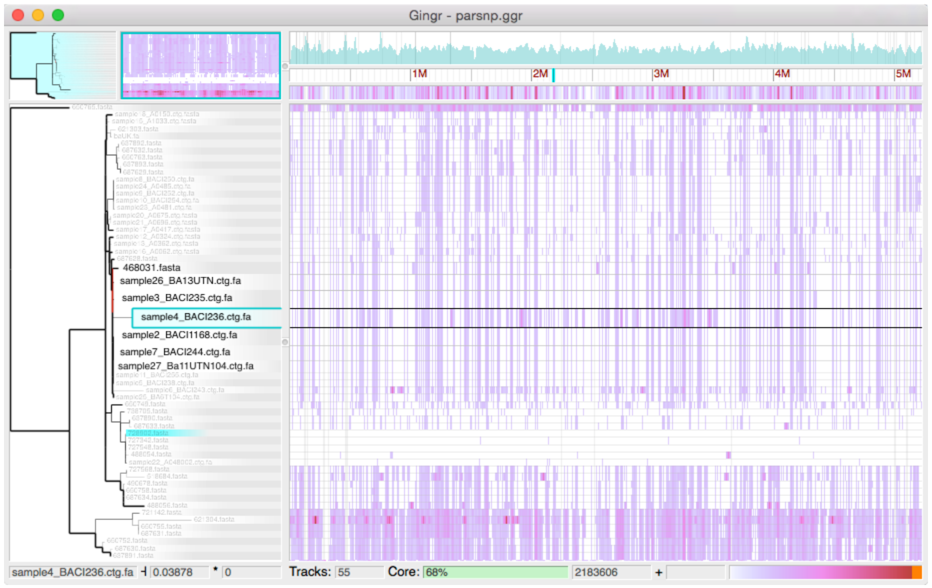
Release 1.0

Brian D. Ondov, Todd J. Treangen, Adam M. Phillippy

Jun 05, 2020

Contents

Harvest is a suite of core-genome alignment and visualization tools for quickly analyzing thousands of intraspecific microbial genomes, including variant calls, recombination detection, and phylogenetic trees.



Tools

- Parsnp - Core-genome alignment and analysis
- Gingr - Interactive visualization of alignments, trees and variants
- HarvestTools - Archiving and postprocessing

Citation

Treangen TJ, Ondov BD, Koren S, Phillippy AM. The Harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes. Genome Biology, 15 (11), 1-15 [PDF]

Download (v1.1.2, 24-Mar-2015)

- Harvest-OSX64-v1.1.2.tar.gz
- Harvest-Linux64-v1.1.2.tar.gz

CHAPTER 1

Parsnp

Rapid core genome multi-alignment

Project home page: https://github.com/marbl/parsnp

Parsnp was designed to align the core genome of hundreds to thousands of bacterial genomes within a few minutes to few hours. Input can be both draft assemblies and finished genomes, and output includes variant (SNP) calls, core genome phylogeny and multi-alignments. Parsnp leverages contextual information provided by multi-alignments surrounding SNP sites for filtration/cleaning, in addition to existing tools for recombination detection/filtration and phylogenetic reconstruction.

Contents:

1.1 Quickstart

Note: If you are currently using a Parsnp version prior to **v1.2** or Harvest version prior to **v1.1.2**, you should update to use the latest release before proceeding. The latest release includes a critical fix for FastTree2; see http://darlinglab.org/blog/2015/03/23/not-so-fast-fasttree.html for further details.

1.1.1 Before you run

1. MUMi based recruitment is useful for quickly identifying clades of closely related genomes from a genomic DB
    - However, it is conservative and can under recruit genomes
    - To force inclusion of all genomes in a given directory, use the *-c* flag.
2. The precompiled binaries are created using Pyinstaller and packaged as a single file:
    - The most likely cause for failure is a lack of free space in the /tmp directory. By default, PyInstaller will search a standard list of directories and sets tempdir to the first one which the calling user can create files in.
    - The list of directories where the archive will be extracted:

- The directory named by the TMPDIR environment variable.
- The directory named by the TEMP environment variable.
- The directory named by the TMP environment variable.

1.1.2 Download, install & run

Parsnp is distributed as a precompiled binary that should be devoid of external dependencies (all included in dist). The three steps below represent the fastest way to start using the software:

On OSX:

1. wget https://github.com/marbl/parsnp/releases/download/v1.2/parsnp-OSX64-v1.2.tar.gz
2. tar -xvf parsnp-OSX64-v1.2.tar.gz

On Linux:

1. wget https://github.com/marbl/parsnp/releases/download/v1.2/parsnp-Linux64-v1.2.tar.gz
2. tar -xvf parsnp-Linux64-v1.2.tar.gz

Basic usage:

From command-line:

```
parsnp -p <threads> -d <directory of genomes> -r <ref genome>
```

Advanced usage:

Parsnp quick start for three example scenarios.

With reference & genbank file:

```
parsnp -g <reference_replicon1,reference_replicon2,..> -d <genome_dir> -p <threads>
```

NOTE:

1. Genbank files are currently expected to have GI numbers for indexing. This means custom Genbank files (not downloaded from NCBI) will not have annotations appear in Gingr, though the alignment should still work. The dependency on GIs is expected to change in future versions.
2. GenBank files can only be specific for the reference genome
3. -g and -r are mutually exclusive; you can either provide a fasta file for your reference genome, or GenBank file, but not both.
4. All non-reference genomes are captured with the -d parameter. These genomes *must* be in fasta format and located within the specified directory.

With reference but without genbank file:

```
parsnp -r <reference_genome> -d <genome_dir> -p <threads>
```

Autorecruit reference to a draft assembly:

```
parsnp -q <draft_assembly> -d <genome_db> -p <threads>
```

Command-line parameters:

Input/output:

```
-c = <flag>: (c)urated genome directory, use all genomes in dir and ignore MUMi?␣
↪(default = NO)
-d = <path>: (d)irectory containing genomes/contigs/scaffolds
-r = <path>: (r)eference genome (set to ! to pick random one from genome dir)
-g = <string>: Gen(b)ank file(s) (gbk), comma separated list (default = None)
-o = <string>: output directory? default [./P_CURRDATE_CURRTIME]
-q = <path>: (optional) specify (assembled) query genome to use, in addition to␣
↪genomes found in genome dir (default = NONE)
```

MUMi:

```
-U = <float>: max MUMi distance value for MUMi distribution
-M = <flag>: calculate MUMi and exit? overrides all other choices! (default: NO)
-i = <float>: max MUM(i) distance (default: autocutoff based on distribution of MUMi␣
↪values)
```

MUM search:

```
-a = <int>: min (a)NCHOR length (default = 1.1*Log(S))
-C = <int>: maximal cluster D value? (default=100)
-z = <path>: min LCB si(z)e? (default = 25)
```

LCB alignment:

```
-D = <float>: maximal diagonal difference? Either percentage (e.g. 0.2) or bp (e.g.␣
↪100bp) (default = 0.12)
-e = <flag> greedily extend LCBs? experimental! (default = NO)
-n = <string>: alignment program (default: libMUSCLE)
-u = <flag>: output unaligned regions? .unaligned (default: NO)
```

Recombination filtration:

```
-x = <flag>: enable filtering of SNPs located in PhiPack identified regions of␣
↪recombination? (default: NO)
```

Misc:

```
-h = <flag>: (h)elp: print this message and exit
-p = <int>: number of threads to use? (default= 1)
-P = <int>: max partition size? limits memory usage (default= 15000000)
-v = <flag>: (v)erbose output? (default = NO)
-V = <flag>: output (V)ersion and exit
```

1.1.3 Output Files

1. Newick formatted core genome SNP tree: $outputdir/parsnp.tree
2. SNPs used to infer phylogeny: $outputdir/parsnp.vcf

3. Gingr formatted binary archive: $outputdir/parsnp.ggr
4. XMFA formatted multiple alignment: $outputdir/parsnp.xmfa

1.1.4 Included external software/packages

- FastTree2 : http://meta.microbesonline.org/fasttree
- Muscle : http://www.drive5.com/muscle
- PhiPack : http://www.maths.otago.ac.nz/~dbryant/software.html

fixme

1.2 Installation from source

1.2.1 Required for building from source:

- 64-bit Linux/*nix or OSX (>= v10.7)
- autoconf && automake && libtool
- gcc (>= v4.2.*)
- OpenMP
- Python (>= 2.6.*)

1.2.2 Build (local)

Please see the Parsnp README. for installation instructions.

1.3 Tutorial

Parsnp download instructions

To further demonstrate the functionality of Parsnp we have prepared two small tutorial datasets. The first dataset is a MERS coronavirus outbreak dataset involving 49 isolates. The second dataset is a selected set of 31 Streptococcus pneumoniae genomes. Both of these datasets should run on modestly equipped laptops in a few minutes.

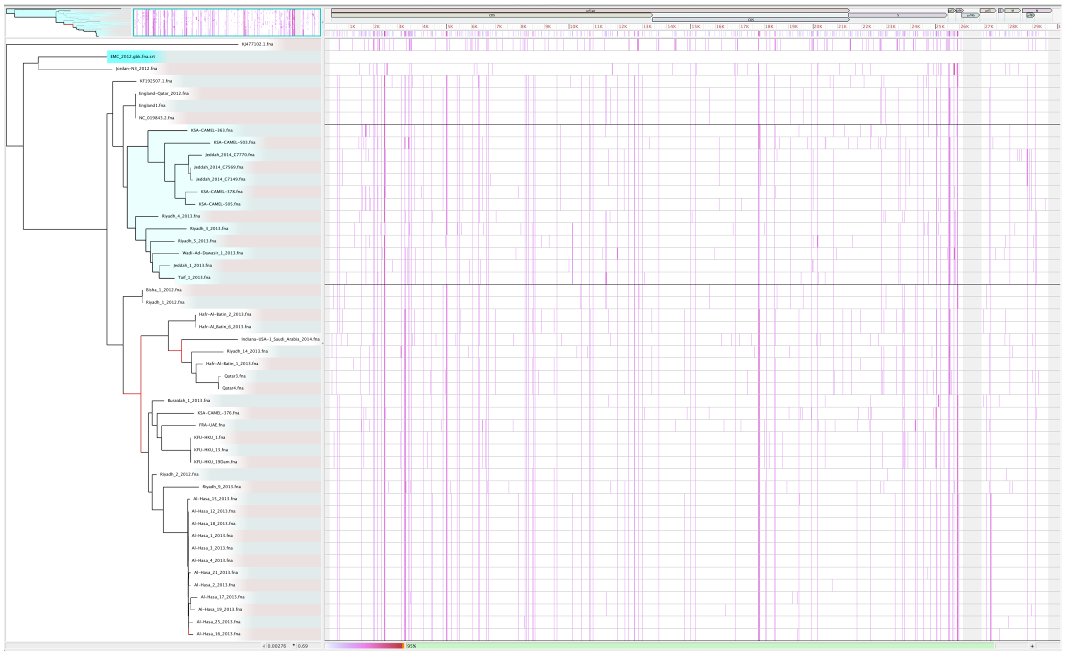
1) 49 MERS Coronavirus genomes
    - Download genomes:
        - `gzipped tarball`
    - Run parsnp with default parameters

```
./parsnp -g ./ref/EMC_2012.gbk -d ./mers49 -c
```

    - Command-line output

```
|--Parsnp v1.0--|
For detailed documentation please see --> http://harvest.readthedocs.org/en/latest
******************************************************************************
SETTINGS:
|-aligner:      libMUSCLE
|-seqdir:       ./mers49
|-outdir:       /Users/treangen/parsnp_tutorial/P_2014_07_21_140147758835
|-OS:           Darwin
|-threads:      8
******************************************************************************
-->Reading Genome (asm, fasta) files from ./mers49..
  |->[OK]
-->Reading Genbank file(s) for reference (.gbk) ./ref/EMC_2012.gbk..
  |->[OK]
-->Running Parsnp multi-MUM search and libMUSCLE aligner..
  |->[OK]
-->Determining repetitive regions..
  |->[OK]
-->Running PhiPack on LCBs to detect recombination..
  |->[SKIP]
-->Reconstructing core genome phylogeny..
  |->[OK]
-->Creating Gingr input file..
  |->[OK]
-->Calculating wall clock time..
  |->Aligned 49 genomes in 0.68 seconds

<<Parsnp finished! All output available in /Users/treangen/parsnp_tutorial/P_2014_07_21_140147758835>>

Validating output directory contents...
        1)parsnp.tree:          newick format tree                      [OK]
        2)parsnp.vcf:           VCF format SNP calls                    [OK]
        3)parsnp.ggr:           harvest input file for gingr (GUI)      [OK]
        4)parsnp.xmfa:          XMFA formatted multi-alignment          [OK]
```

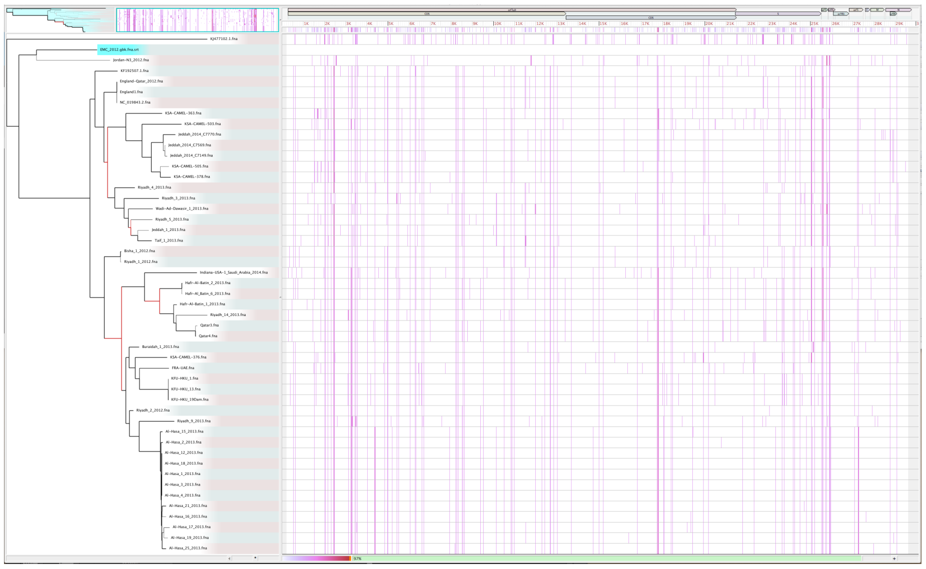
- Visualize with Gingr `GGR`



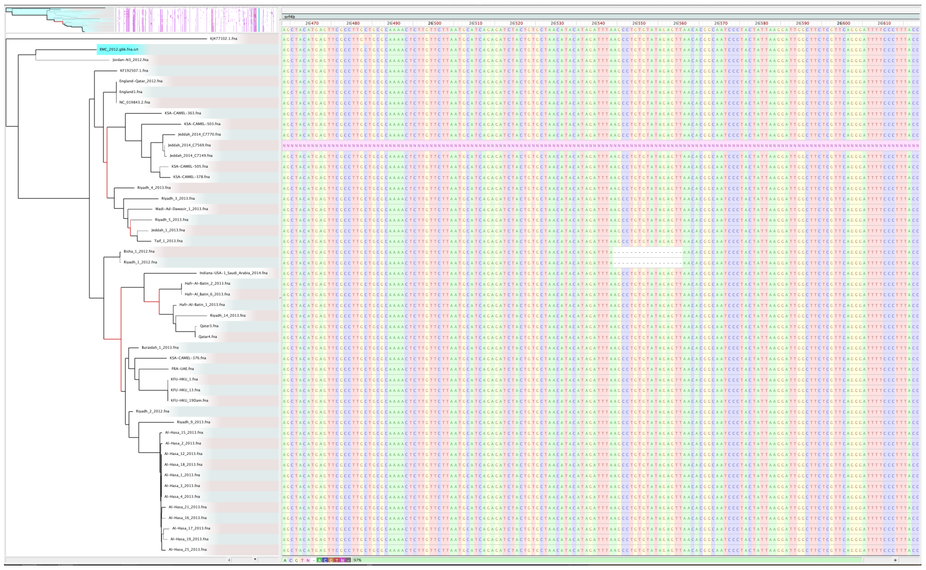
- Configure parameters
  - 95% of the reference is covered by the alignment. This is <100% mainly due to a 1kbp unaligned region from 26kbp to 27kbp.
  - To force alignment across large collinear regions, use the *-C* maximum distance between two collinear MUMs:

```
./parsnp -g ./ref/EMC_2012.gbk -d ./mers49 -C 1000 -c
```

- Visualize again with Gingr `GGR`
  - By adjusting the *-C* parameter, this region is no longer unaligned, boosting the reference coverage to 97%.



- Zoom in with Gingr for nucleotide view of region
  - On closer inspection, a large stretch of N's in Jeddah isolate C7569 was the culprit



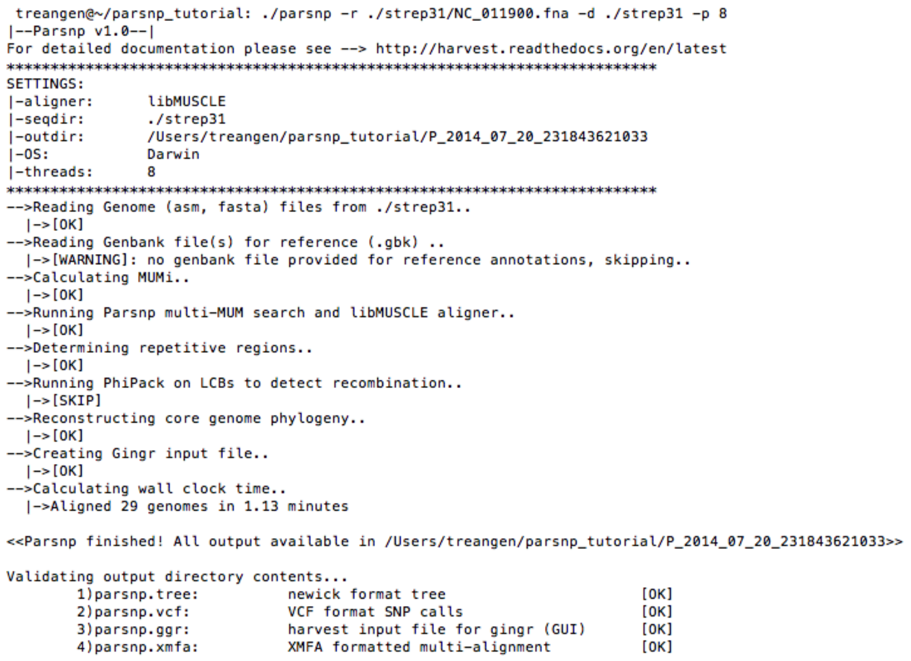
- Inspect Output:
  - Multiple alignment: `XMFA`
  - SNPs: `VCF`
  - Phylogeny: `Newick`

2) 31 Streptococcus pneumoniae genomes
   - Download genomes:
     - `gzipped tarball`

- Run parsnp

```
./parsnp -r ./strep31/NC_011900.fna -d ./strep31 -p <num threads>
```

- Command-line output:

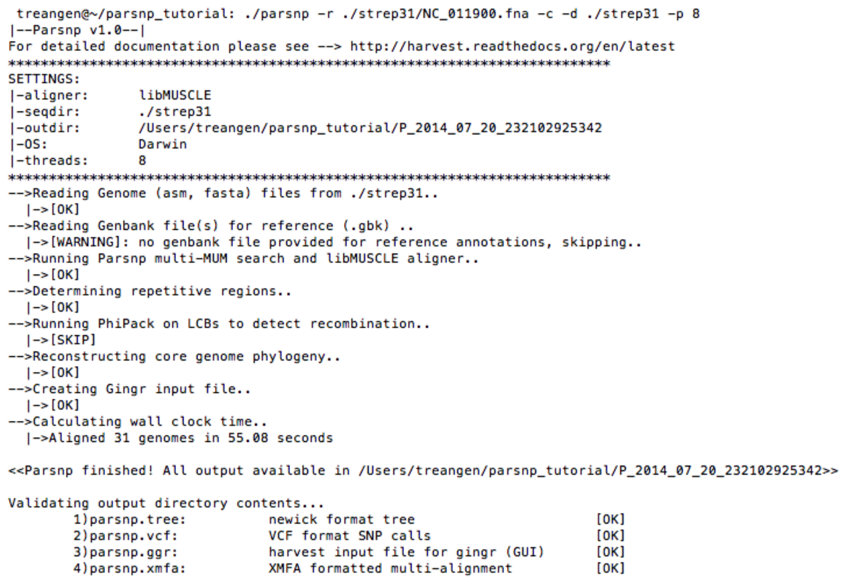


```
 treangen@~/parsnp_tutorial: ./parsnp -r ./strep31/NC_011900.fna -d ./strep31 -p 8
|--Parsnp v1.0--|
For detailed documentation please see --> http://harvest.readthedocs.org/en/latest
**************************************************************************
SETTINGS:
|-aligner:      libMUSCLE
|-seqdir:       ./strep31
|-outdir:       /Users/treangen/parsnp_tutorial/P_2014_07_20_231843621033
|-OS:           Darwin
|-threads:      8
**************************************************************************
-->Reading Genome (asm, fasta) files from ./strep31..
  |->[OK]
-->Reading Genbank file(s) for reference (.gbk) ..
  |->[WARNING]: no genbank file provided for reference annotations, skipping..
-->Calculating MUMi..
  |->[OK]
-->Running Parsnp multi-MUM search and libMUSCLE aligner..
  |->[OK]
-->Determining repetitive regions..
  |->[OK]
-->Running PhiPack on LCBs to detect recombination..
  |->[SKIP]
-->Reconstructing core genome phylogeny..
  |->[OK]
-->Creating Gingr input file..
  |->[OK]
-->Calculating wall clock time..
  |->Aligned 29 genomes in 1.13 minutes

<<Parsnp finished! All output available in /Users/treangen/parsnp_tutorial/P_2014_07_20_231843621033>>

Validating output directory contents...
        1)parsnp.tree:          newick format tree                      [OK]
        2)parsnp.vcf:           VCF format SNP calls                    [OK]
        3)parsnp.ggr:           harvest input file for gingr (GUI)      [OK]
        4)parsnp.xmfa:          XMFA formatted multi-alignment          [OK]
```

- Force inclusion of all genomes (-c)

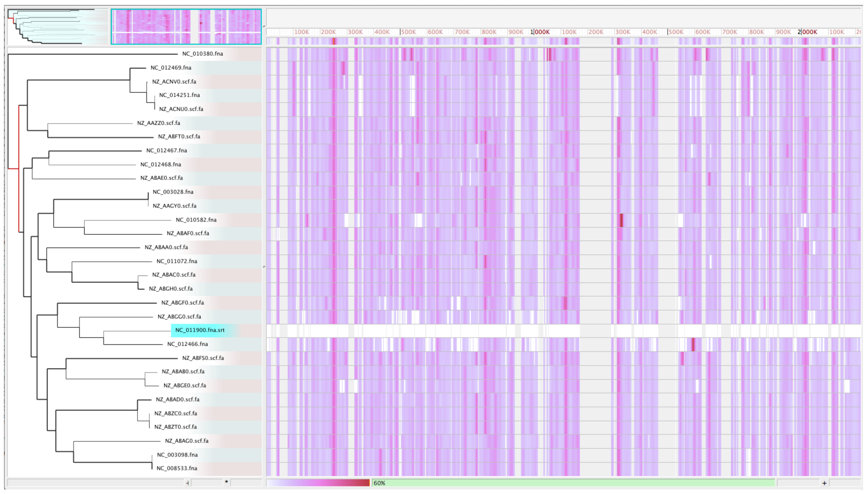
```
./parsnp -r ./strep31/NC_011900.fna -d ./strep31 -p <num threads> -c
```

- Command-line output:



```
 treangen@~/parsnp_tutorial: ./parsnp -r ./strep31/NC_011900.fna -c -d ./strep31 -p 8
|--Parsnp v1.0--|
For detailed documentation please see --> http://harvest.readthedocs.org/en/latest
**************************************************************************
SETTINGS:
|-aligner:      libMUSCLE
|-seqdir:       ./strep31
|-outdir:       /Users/treangen/parsnp_tutorial/P_2014_07_20_232102925342
|-OS:           Darwin
|-threads:      8
**************************************************************************
-->Reading Genome (asm, fasta) files from ./strep31..
  |->[OK]
-->Reading Genbank file(s) for reference (.gbk) ..
  |->[WARNING]: no genbank file provided for reference annotations, skipping..
-->Running Parsnp multi-MUM search and libMUSCLE aligner..
  |->[OK]
-->Determining repetitive regions..
  |->[OK]
-->Running PhiPack on LCBs to detect recombination..
  |->[SKIP]
-->Reconstructing core genome phylogeny..
  |->[OK]
-->Creating Gingr input file..
  |->[OK]
-->Calculating wall clock time..
  |->Aligned 31 genomes in 55.08 seconds

<<Parsnp finished! All output available in /Users/treangen/parsnp_tutorial/P_2014_07_20_232102925342>>

Validating output directory contents...
        1)parsnp.tree:          newick format tree                      [OK]
        2)parsnp.vcf:           VCF format SNP calls                    [OK]
        3)parsnp.ggr:           harvest input file for gingr (GUI)      [OK]
        4)parsnp.xmfa:          XMFA formatted multi-alignment          [OK]
```

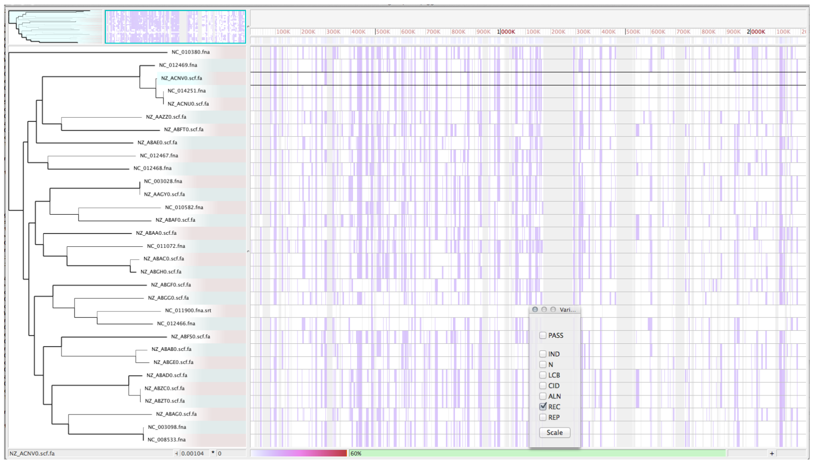
- Visualize with Gingr `GGR`



- Enable recombination detection/filter (-x)

```
./parsnp -r ./strep31/NC_011900.fna -d ./strep31 -p <num threads> -c -x
```

- Re-visualize with Gingr `GGR`
  - Bootstrap values have improved after running recombination filter; columns with filtered SNPs are displayed in image:



- Inspect Output:
  - Multiple alignment: `XMFA`
  - SNPs: `VCF`
  - Phylogeny: `Newick`

1.4 FAQs

Q. **What is Parsnp ?**

A. Parsnp takes both draft and finished genomes of closely related strains as input, performs conservative core genome alignment and as output returns multi-alignments (XMFA), variants (VCF), core genome phylogeny (Newick) and Gingr input format (GGR).

Q. **Why should I use Parsnp?**

A. The main advantages of Parsnp over alternative approaches is robust filtration of variant (SNP) calls, multiple alignments as output and superior speed. Parsnp can align 200-300 bacterial strains in <30 minutes on a 16-core server and ~1000 in a couple of hours.

Q. **Why should I *not* use Parsnp?**

A. If you are interested in pan genome/whole genome alignment, existing tools for the job that perform well include Mauve, Mugsy, among others. In addition, Parsnp is tailored for intraspecific genome analysis (outbreak analysis of a pathogen, etc). One main limitation of Parsnp is that it cannot handle subsets (core genome only) and is not as sensitive as existing methods.

Q. **How can I visualize the results?**

A. Gingr (http://github.com/marbl/gingr) can open Parsnp output and provide an interactive display of multi-alignments, variants and the phylogenetic tree estimated from the core genome alignment.

Q. **What % of genome X is aligned? What is the core genome alignment size?**

A. Within the log output, there are coverage values listed that individually indicate the percentage of a given genome that is included in the core genome alignment. Note, this includes the Muscle aligned-regions plus the maximal unique matches (MUMs). The core genome alignment size can then be calculated by multiplying the coverage value, for a given genome, by its length.

Q. **Only a small percentage (<40%) of the reference genome covered by the alignments, huh?**

A. Parsnp is a conservative core genome alignment method that necessarily requires that all genomes are present in each aligned regions. The focus is on aligning 1000s of closely related bacterial strains quickly while maintaining sensitivity comparable to existing WGA methods. In additon, the core genome has been shown to contain as few as 30-40% of the gene content (even in very closely-related clades) due to reductive genome evolution and/or a large accessory genome (with plenty of IS/phage elements). However, for increased sensitivity w.r.t aligned regions, and alignments containing subsets, both Mugsy and Mauve are terrific tools for the job.

Q. **I am not sure which reference genome to choose, help!**

A. Since the core necessarily includes all genomes, the choice of reference does not matter (with a couple important exceptions, continue reading). Feel free to use the parameter '-r !' to randomly select a reference if you are feeling indecisive or if all of the genomes contained in the genome directory are of similar quality. Typically, finished/closed genomes are used a the reference strain to ensure they are high-quality and do not contain assembly artifacts or contaminant.

Q. **How are the genomes selected from the input directory? Not all of the genomes I wanted included are in the tree!**

A. By default, parsnp calculates the MUMi distance between the reference and each of the genomes in the genome directory. All genomes with MUMi distance <= 0.01 are included, all others are discarded. To force all genomes present in the genome dir to be included simply include '-c' as a command-line parameter.

Q. **Parsnp fails to report a SNP position output by method X, why is this?**

A. The goal of parsnp is to capture all informative signals found in the core genome of the specified clade of interest. Any SNPs in regions not shared by all genomes will not reported. Additionally, any SNP found in a likely poorly aligned region would also be discarded. Finally, parsnp does not perform LCB extension and therefore may miss SNPs appearing at the end of locally conserved blocks or clusters.

fixme

fixme

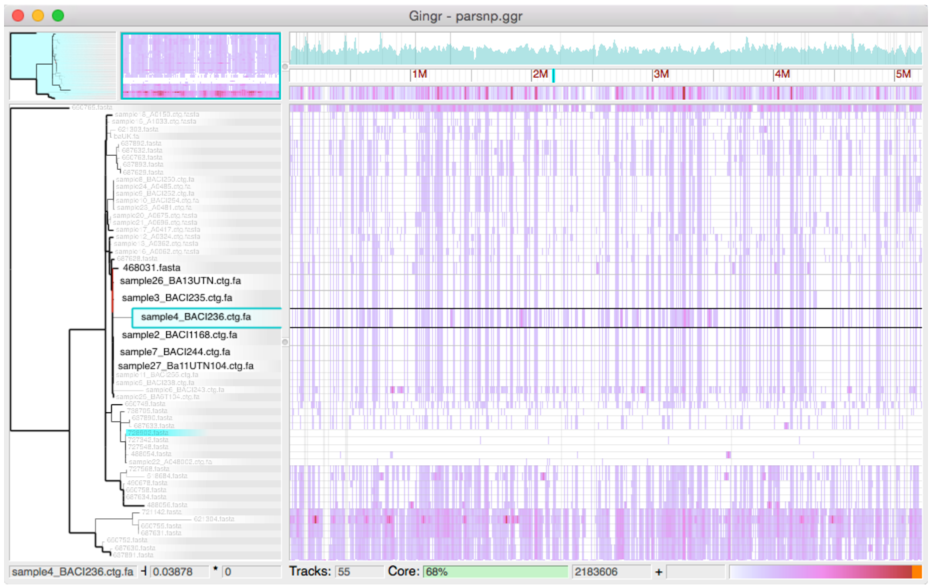
fixme

CHAPTER 2

Gingr

Interactive visualization of alignments, trees and variants

Gingr is an interactive tool for exploring large-scale phylogenies in tandem with their corresponding multi-alignments. Gingr can display informative overviews for hundreds or thousands of genomes, while allowing researchers to move quickly to more detailed views of specific subclades and genomic regions, even down to the nucleotide level of their multi-alignments. Additionally, its dynamic display of variants allows interactive selection of various filters, such as indels, poorly aligned regions and suspected sites of recombination. Gingr works chiefly in tandem with Parsnp, an efficient tool for core-genome multi-alignment and phylogenetic reconstruction. It is also applicable, however, to other analytical tools, accepting standard file formats such as multi-Fasta, XMFA, Newick and VCF.



Download (v1.3)

- gingr-OSX64-v1.3.zip
- gingr-Linux64-v1.3.tar.gz

Documentation

2.1 Requirements

2.1.1 Mac

- OS X 10.7 (Lion) or later (requires 64 bit architecture)

2.1.2 Linux

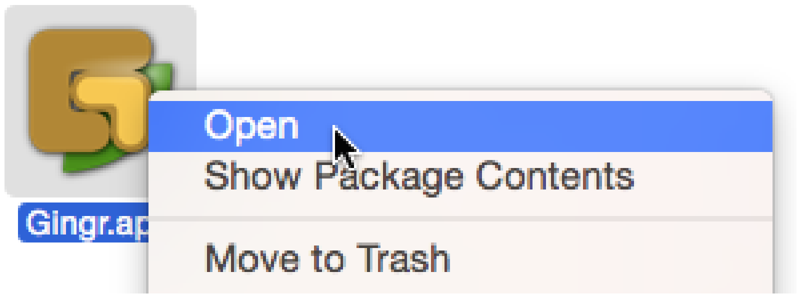
- 64 bit architecture
- Common distributions
  - The Gingr binary should work with most recent (within ~5 years) versions of common Linux distributions, e.g.:
  - CentOS (6+)
  - Ubuntu (9+)
  - Fedora (10+)
  - . . . and many others
- Source
- If the Gingr binary does not work on a particular distribution, it may be possible to build from source
- gcc 4.8+ is required for building

2.2 Tutorial

2.2.1 Running Gingr

Mac OS X

- Gingr.app can be moved to the Applications folder if desired
- Double-click Gingr.app to run
- Depending on your security settings, there may be an error that Gingr is not from the Mac App Store or is from an unidentified developer. To run it anyway:
- Right click on Gingr.app
- Select “Open” from the menu
- Click the “Open” button at the next prompt

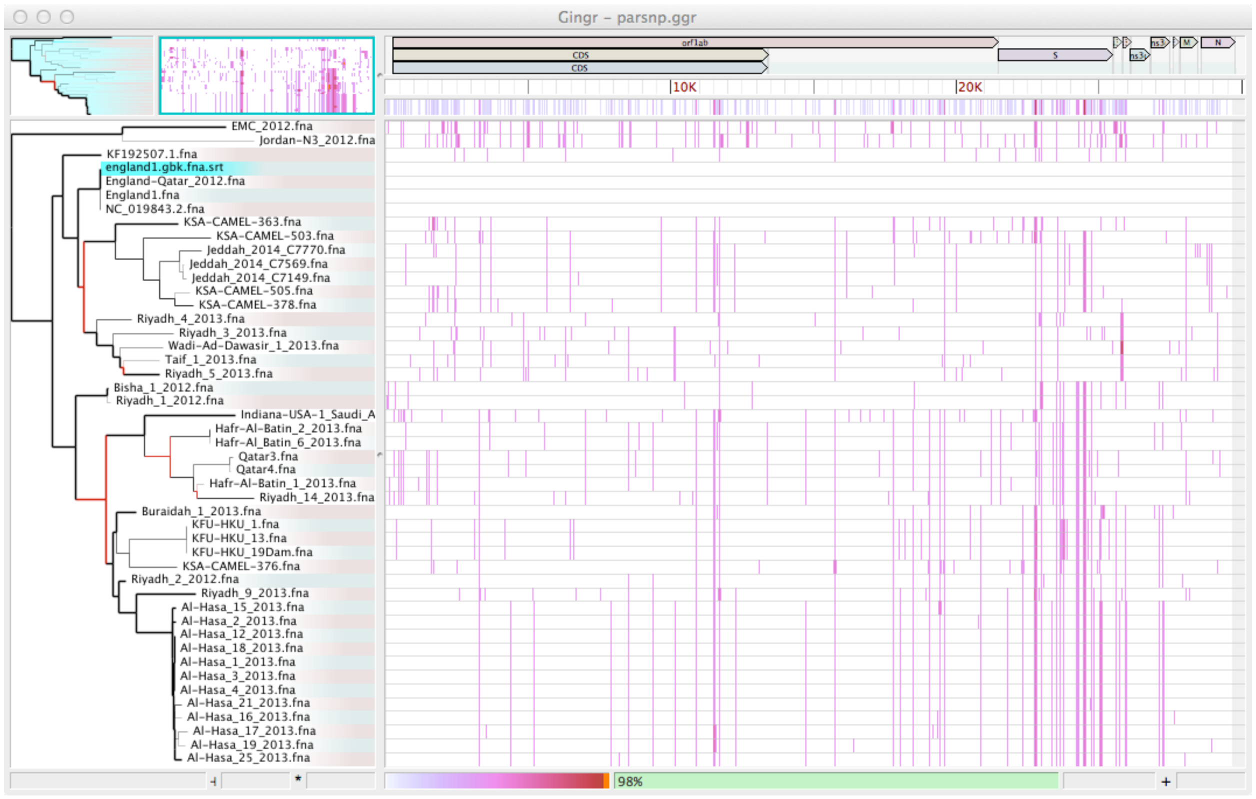


Linux

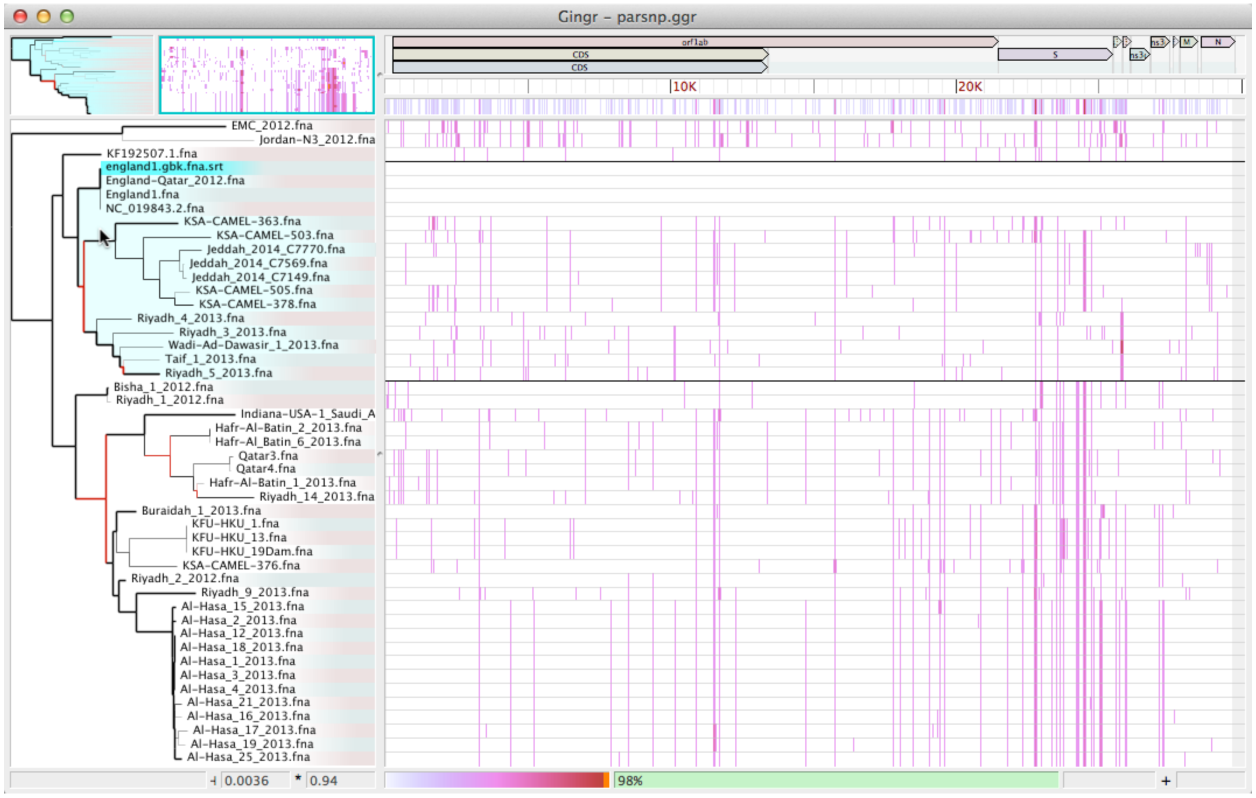
- From the desktop
- Click on the “gingr” binary
- From a terminal
- Navigate to the folder with the “gingr” binary
- Run “./gingr”

2.2.2 Browsing a Gingr file

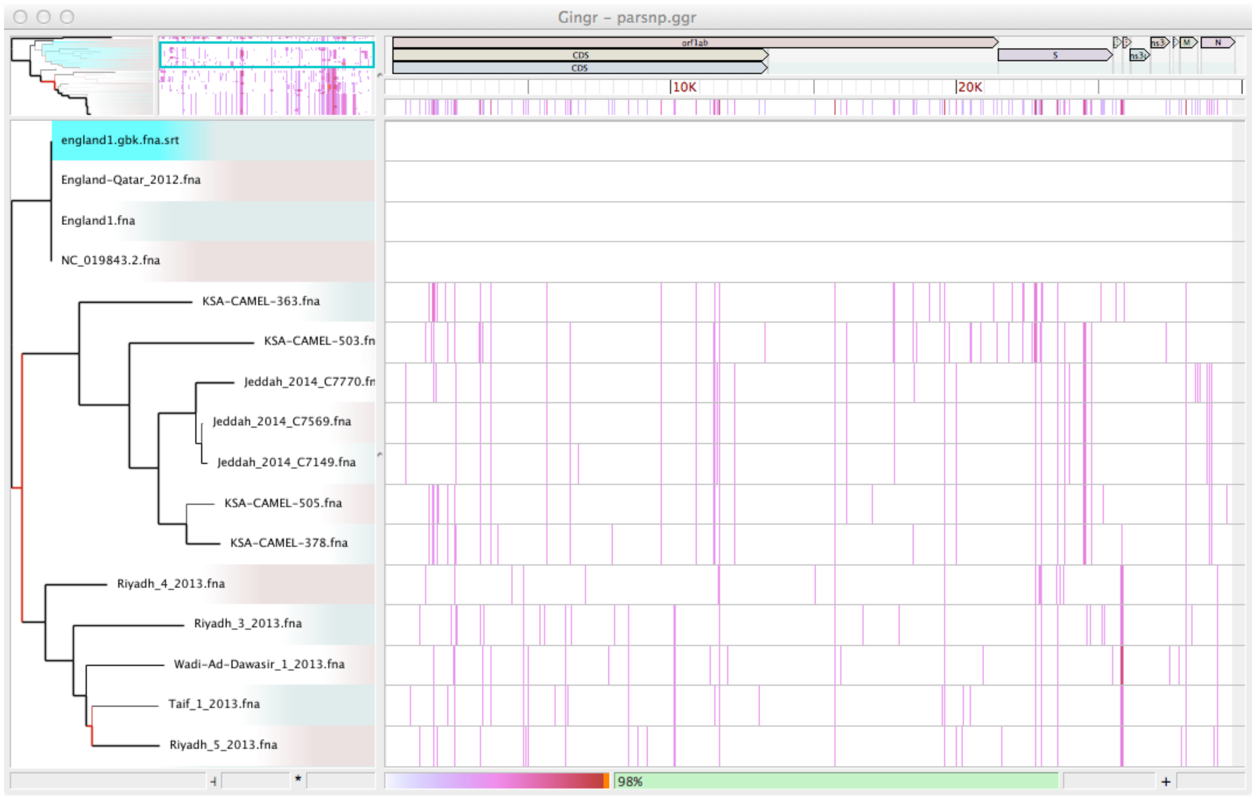
- Download `Gingr input file`
- Open in Gingr (File->Open)



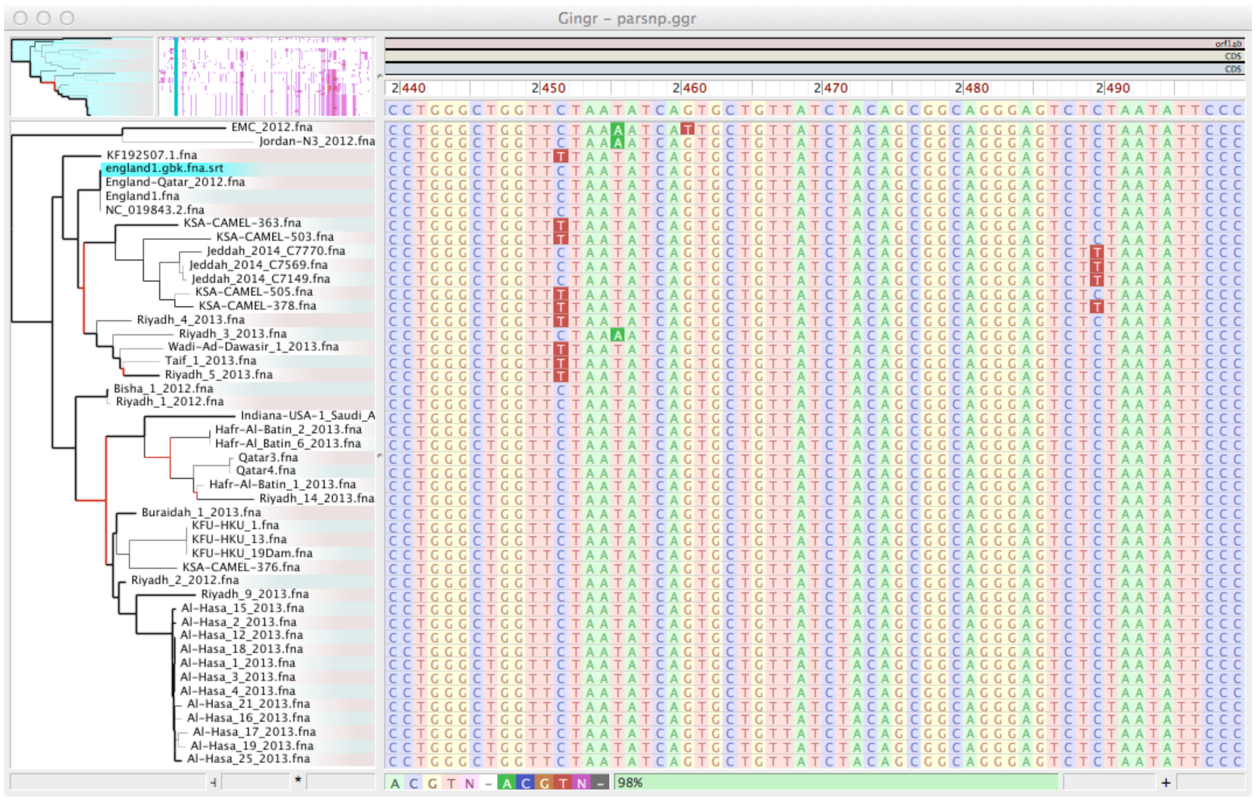
- The phylogeny appears on the left. Hover over a clade to highlight and outline the corresponding tracks to the right.



- Click to zoom in on the clade

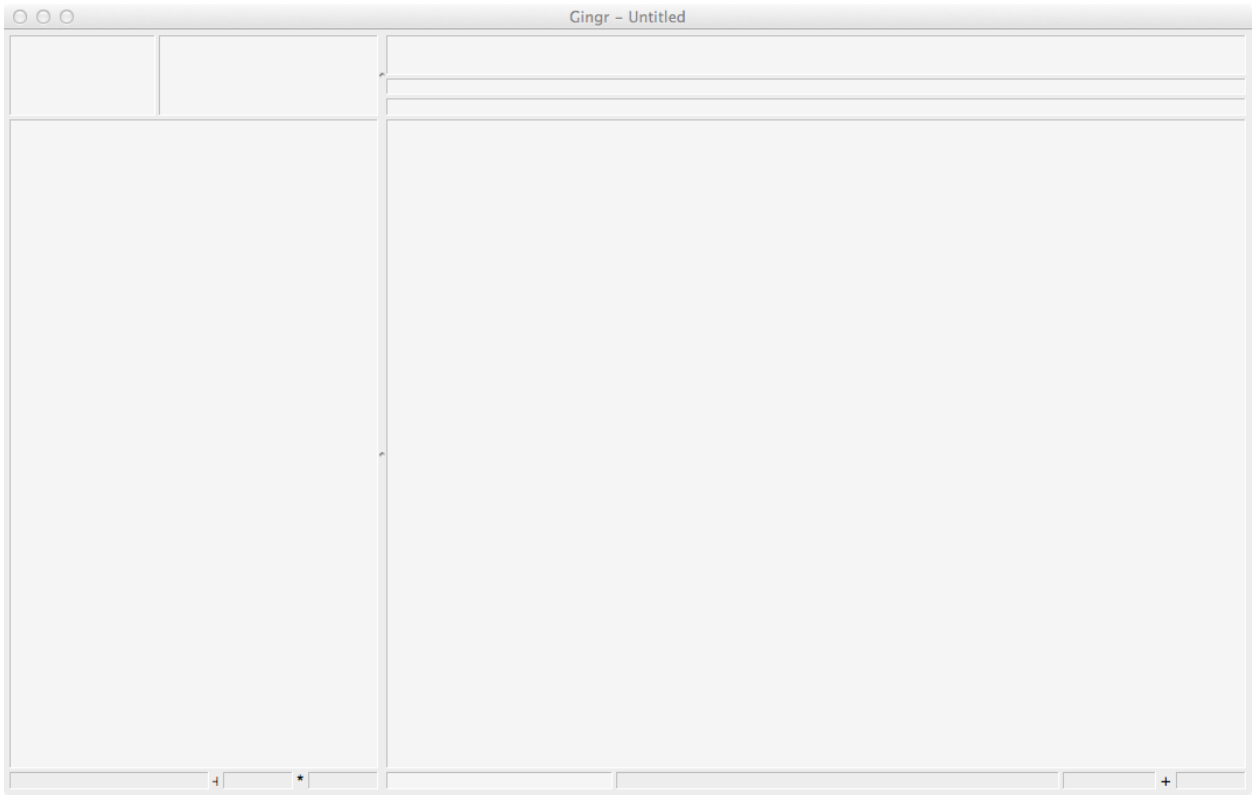


- The multiple alignment appears on the right, shown as a SNP heatmap when zoomed out. To see the full alignment, zoom in with the mouse wheel or by selecting a region in the ruler.

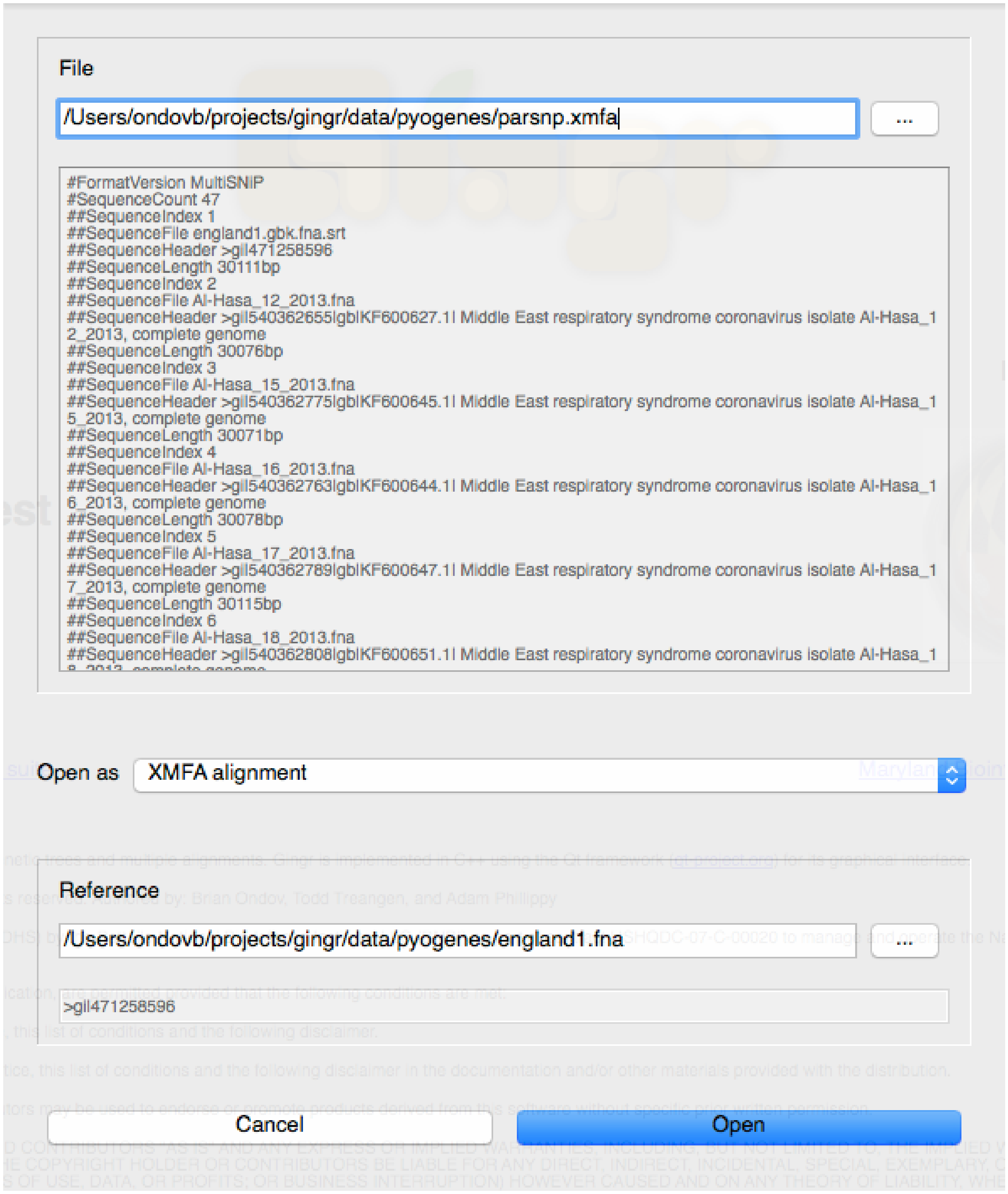


2.2.3 Importing other files

- Create a new workspace (File->New)



- Download the data files
  - Alignment: `xmfa`
  - Reference: `fasta`
  - Annotations: `genbank`
  - Phylogeny: `newick`
- Open the XMFA alignment (File->Open). Since XMFA files can be accompanied by reference files, the Open dialog will appear. Choose the Fasta file as the reference in this window.



/Users/ondovb/projects/gingr/data/pyogenes/parsnp.xmfa

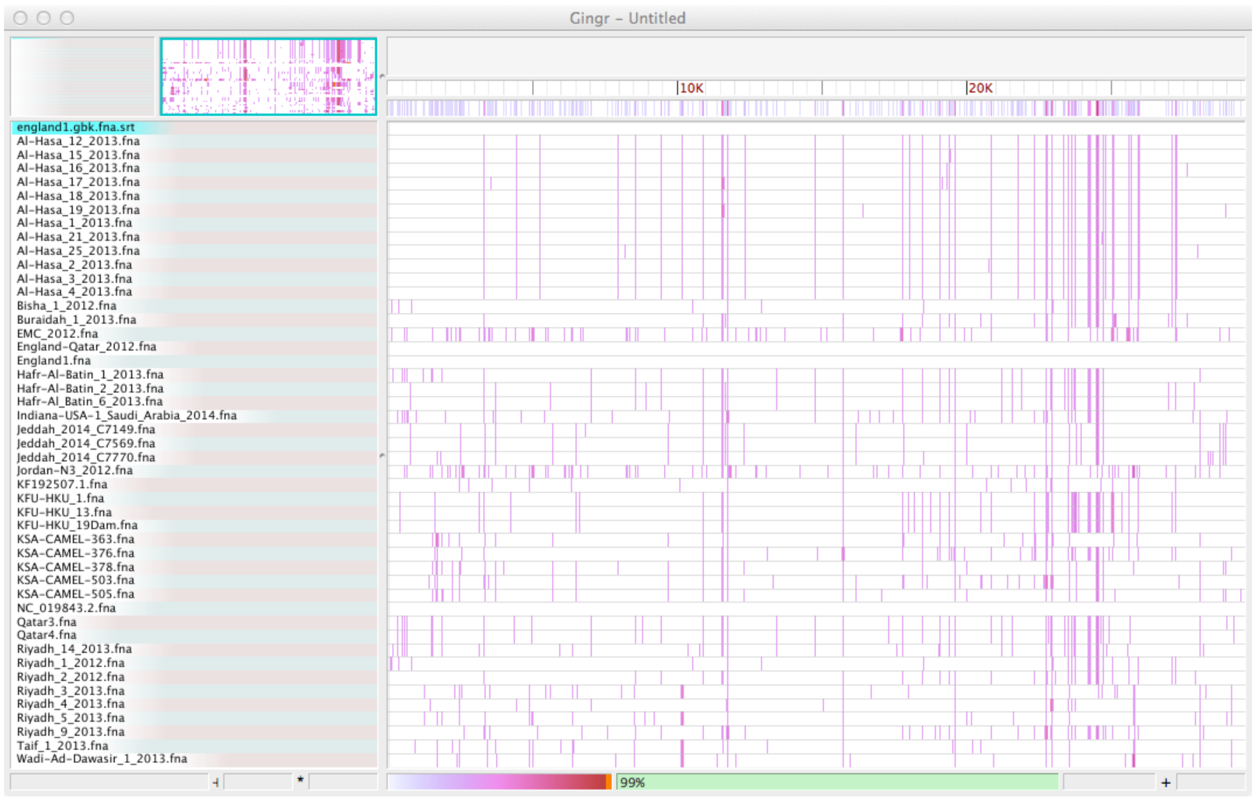
```
#FormatVersion MultiSNIP
#SequenceCount 47
##SequenceIndex 1
##SequenceFile england1.gbk.fna.srt
##SequenceHeader >gi|471258596
##SequenceLength 30111bp
##SequenceIndex 2
##SequenceFile Al-Hasa_12_2013.fna
##SequenceHeader >gi|540362655|gb|KF600627.1| Middle East respiratory syndrome coronavirus isolate Al-Hasa_1
2_2013, complete genome
##SequenceLength 30076bp
##SequenceIndex 3
##SequenceFile Al-Hasa_15_2013.fna
##SequenceHeader >gi|540362775|gb|KF600645.1| Middle East respiratory syndrome coronavirus isolate Al-Hasa_1
5_2013, complete genome
##SequenceLength 30071bp
##SequenceIndex 4
##SequenceFile Al-Hasa_16_2013.fna
##SequenceHeader >gi|540362763|gb|KF600644.1| Middle East respiratory syndrome coronavirus isolate Al-Hasa_1
6_2013, complete genome
##SequenceLength 30078bp
##SequenceIndex 5
##SequenceFile Al-Hasa_17_2013.fna
##SequenceHeader >gi|540362789|gb|KF600647.1| Middle East respiratory syndrome coronavirus isolate Al-Hasa_1
7_2013, complete genome
##SequenceLength 30115bp
##SequenceIndex 6
##SequenceFile Al-Hasa_18_2013.fna
##SequenceHeader >gi|540362808|gb|KF600651.1| Middle East respiratory syndrome coronavirus isolate Al-Hasa_1
```

Open as: XMFA alignment

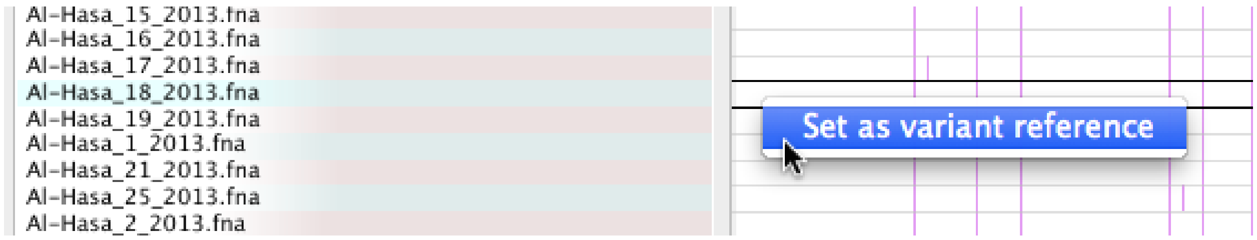
Reference: /Users/ondovb/projects/gingr/data/pyogenes/england1.fna

>gi|471258596

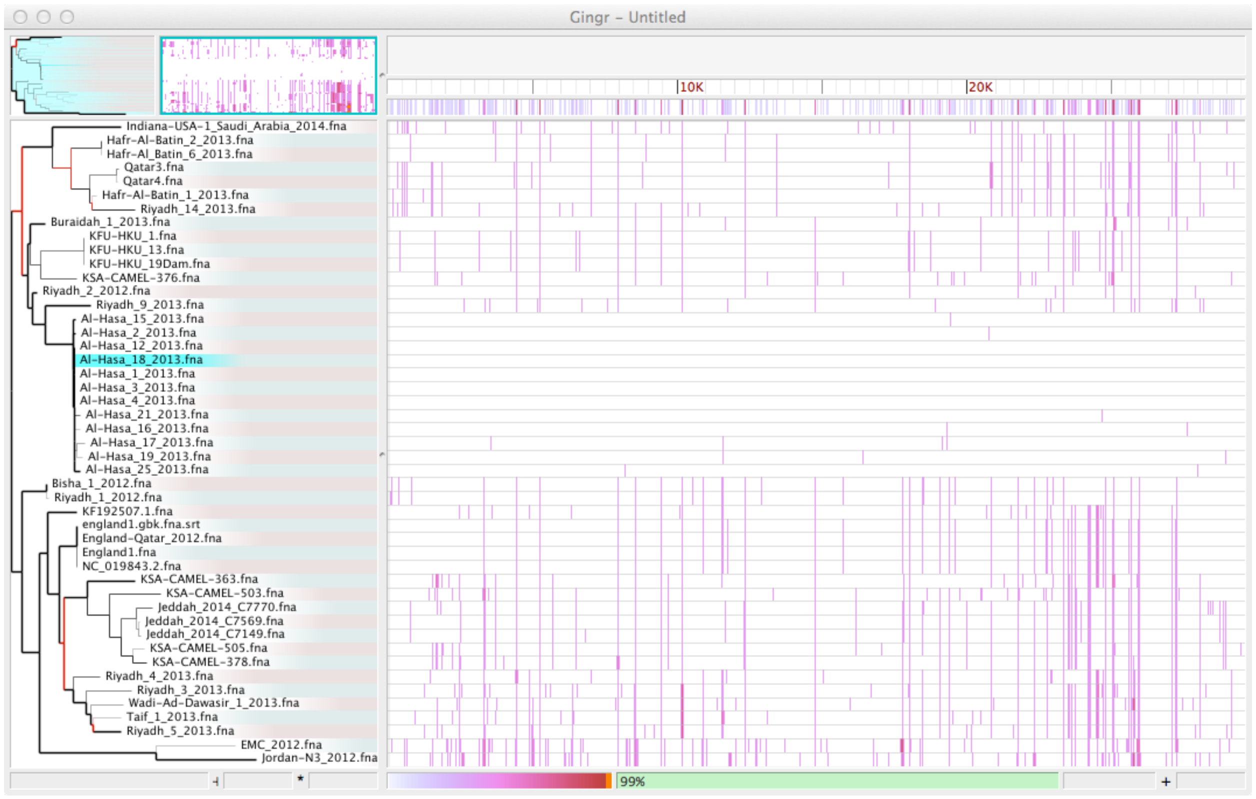
- The preview panes allow you to ensure that the header for the reference is the same as the first sequence in the XMFA. This allows sequences between LCBs to be shown and allows annotations to be added later.



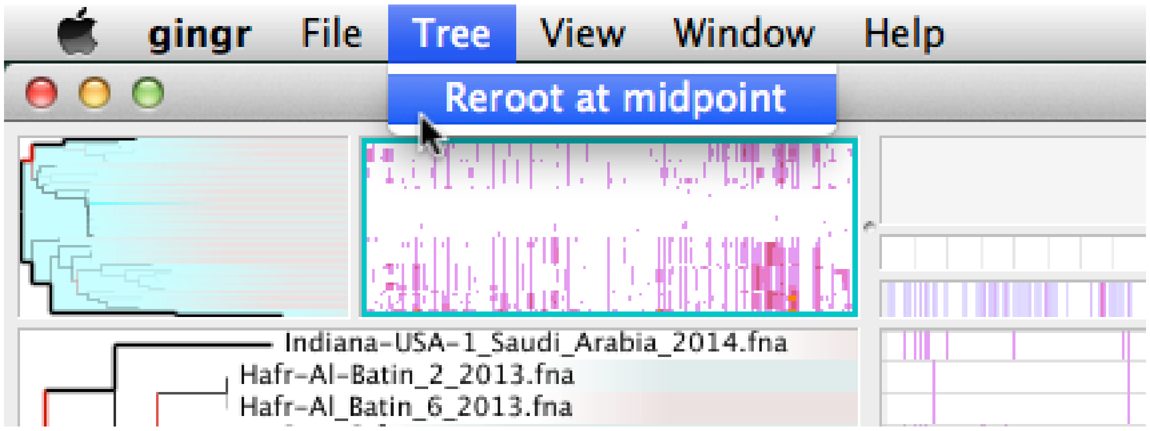
- The track highlighted in blue ("england.gbk.fna.srt") is the current reference for variants. Select a new reference by right-clicking on a track.



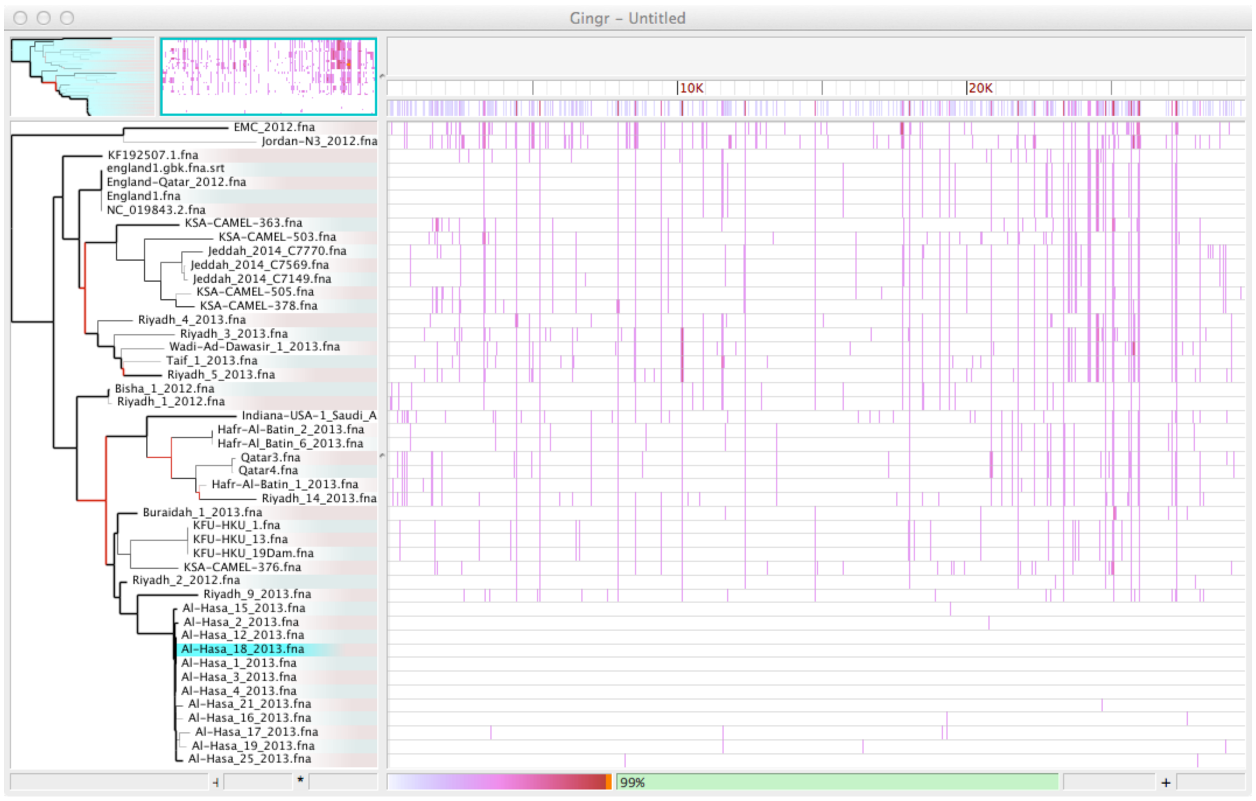
- Next, import the phylogenetic tree (File->Open)



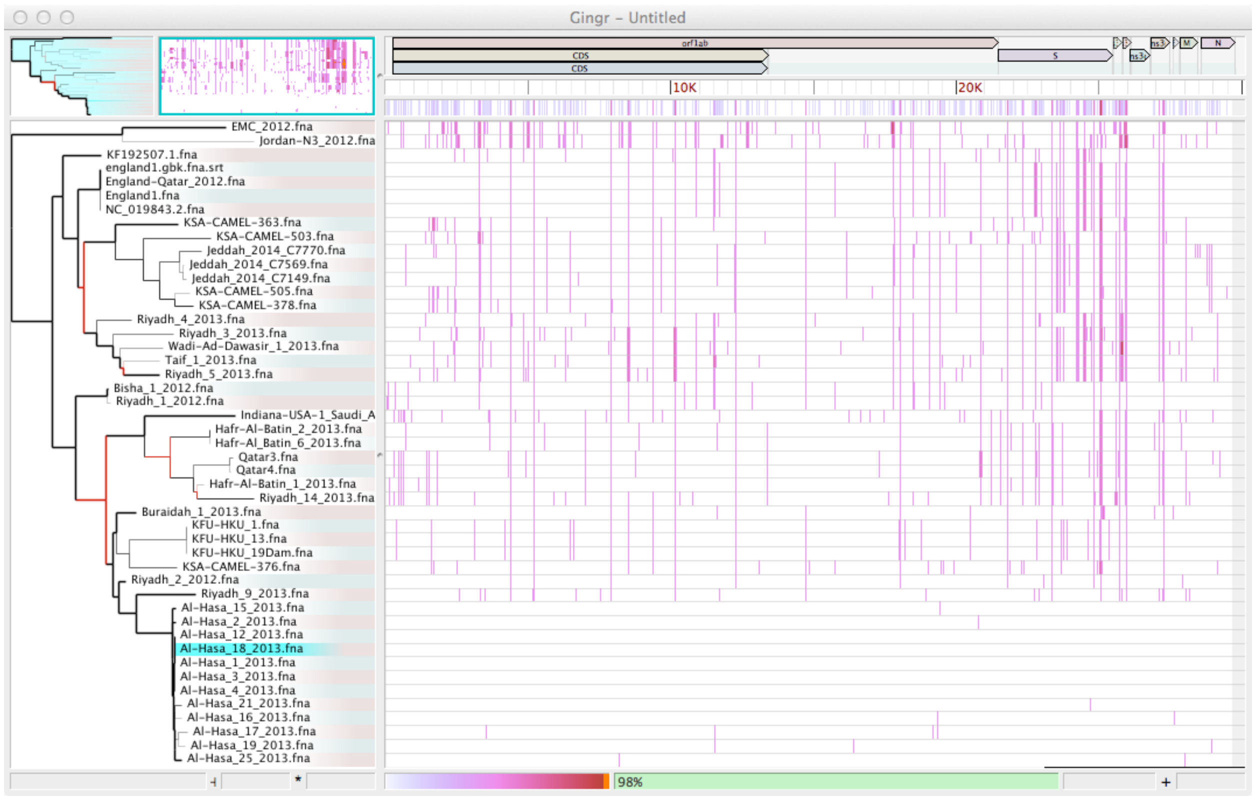
- Reroot the tree at the midpoint (Tree->Reroot at midpoint)



- The tree will now be balanced at the center of the longest path



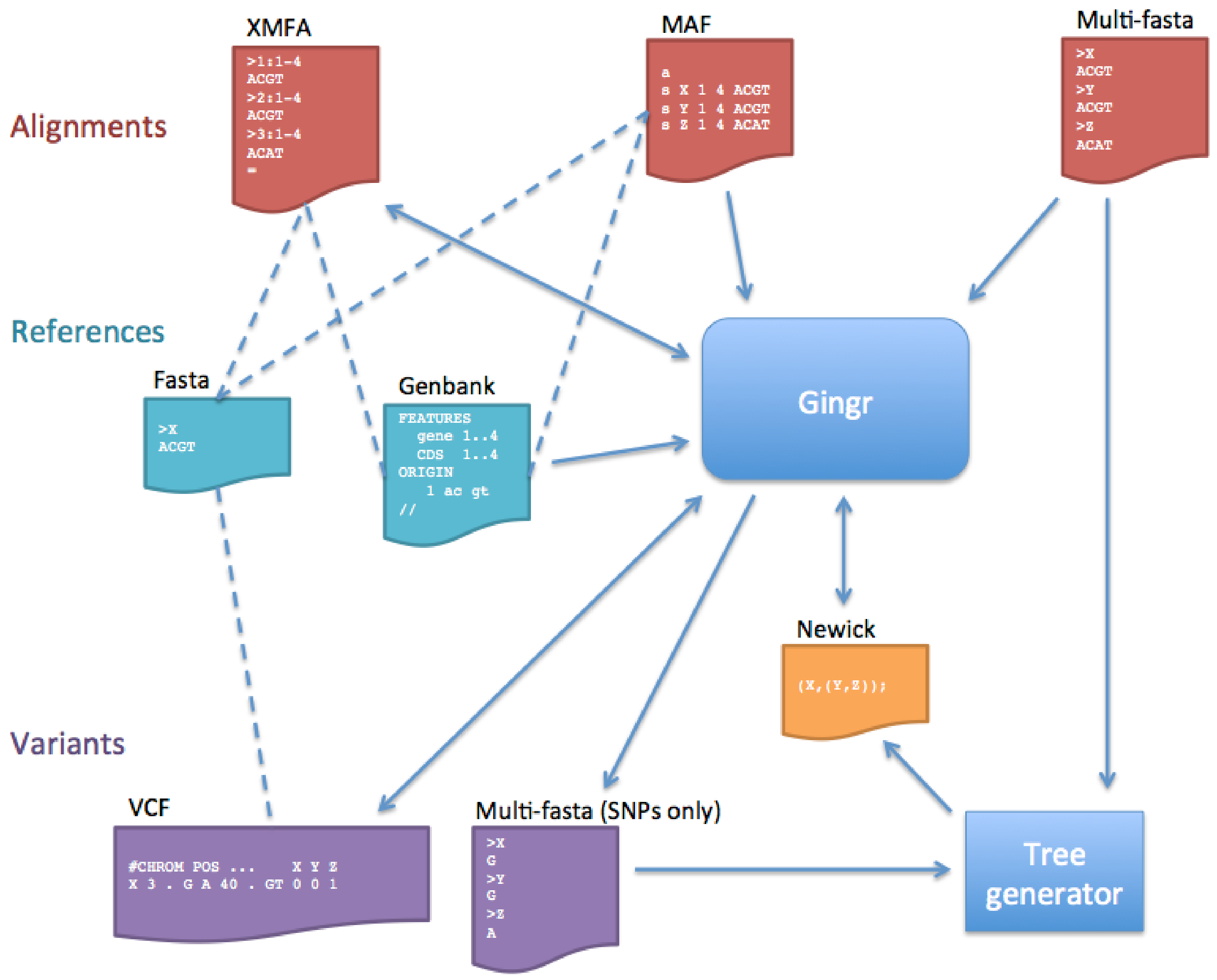
- Finally, import the annotations (File->Open)



- The workspace can be saved to share or return to later (File->Save)

2.3 File formats

The flowchart below describes the various file formats that can be imported or exported to/from Gingr (or the *harvest-tools* command line utility).



- **Alignments**
  - Core only: The Gingr file format stores core alignments, or alignments that involve all genomes. When alignments are loaded, blocks that are not core will be discarded.
  - When importing MAF alignments, the first sequence of the first core block is used as the reference. Each reference contig will be padded with Ns up to its first LCB and between subsequent LCBs. If alignment blocks overlap in reference coordinate space, the block seen earlier in the file will be kept; the later one will be ignored.
  - Multi-fasta: This format does not store rearrangement information, so the alignment is treated as a single LCB.
- **References**
  - XMFA files can be accompanied by Fasta reference files to provide sequence between LCBs and to allow Genbank annotations (which must have matching GI numbers) to be loaded later. Genbank files that contain sequence can also be used as references.

  - Multi-fasta alignments will use the first sequence as the reference. Genbank annotations can be loaded later if the GIs match.

- **Variants**
  - VCF files must be imported with a Fasta reference. The only fields imported are sequence identifier (CHROM), position (POS), reference allele (REF), alternate alleles (ALT), quality (QUAL), filters (FILTER, including ##FILTER specifications in the header), and genotype (GT); all other information is ignored. Additionally, Since VCF does not store complete alignment information, any insertions larger than one base will be replaced by an LCB boundary when importing. If a genotype is diploid or polyploid, only the first haplotype is used in the multi-alignment (the others are ignored). Symbolic alleles and breakends are currently unsupported and will also be ignored. When writing to VCF, only the imported fields will be populated. Indel output is also currently unimplemented, so indels will be skipped when writing.
  - The multi-fasta SNP output is the same format as multi-fasta alignments, but only contains columns with unfiltered ("PASS") variants (like a Mauve SNP file). This is useful for generating phylogenetic trees, but does not contain positional information or rearrangements.

Resources

All releases ~ Source code ~ Report an issue

CHAPTER 3

HarvestTools

Archiving and postprocessing

HarvestTools is a utility for creating and interfacing with Gingr files, which are efficient archives that the Harvest Suite uses to store reference-compressed multi-alignments, phylogenetic trees, filtered variants and annotations. Though designed for use with Parsnp and Gingr, HarvestTools can also be used for generic conversion between standard bioinformatics file formats.

Download (v1.2)

- harvesttools-OSX64-v1.2.zip
- harvesttools-Linux64-v1.2.tar.gz

Documentation

3.1 Quickstart

3.1.1 Before you run

1. harvest-tools VCF outputs indels in non standard format:
    - currently column based, not row based. excluding indel rows (default behavior) converts file into valid VCF format.
    - this will be updated in future version
2. Genbank annotation input:
    - Multiple Genbank files must be specified with multiple *-g* parameters
    - In addition, genome identifier (gi) must match the header of the reference fasta file

3.1.2 Download, install & run

harvest-tools is distributed as a precompiled binary. The three steps below represent the fastest way to start using the software:

On OSX:

1. wget https://github.com/marbl/harvest-tools/releases/download/v1.2/harvesttools-OSX64-v1.2.zip
2. tar -xvf harvesttools-OSX64-v1.2.tar.gz

On Linux:

1. wget https://github.com/marbl/harvest-tools/releases/download/v1.2/harvesttools-Linux64-v1.2.tar.gz
2. tar -xvf harvesttools-Linux64-v1.2.tar.gz

Basic usage:

From command-line:

```
harvesttools -x <input xmfa> -f <input reference fasta> -g <reference␣
↪genbank formatted annotations> -n <newick formatted tree>
```

Advanced usage:

harvest-tools quick start for three example scenarios.

With reference & genbank file as input:

```
harvesttools -g <reference_genbank_file1> -r <reference fasta file> -x <XMFA file> -o␣
↪hvt.ggr
```

With harvest-tools file as input, XMFA output:

```
harvesttools -i input.ggr -X output.xmfa
```

With harvest-tools file as input, fasta formatted SNP file as output:

```
harvesttools -i input.ggr -S output.snps
```

Command-line parameters:

- -i <Gingr input>
- -b <bed filter intervals>,<filter name>,"<description>"
- -B <output backbone intervals>
- -f <reference fasta>
- -F <reference fasta out>
- -g <reference genbank>

- -a <MAF alignment input>
- -m <multi-fasta alignment input>
- -M <multi-fasta alignment output (concatenated LCBs)>
- -n <Newick tree input>
- -N <Newick tree output>
- –midpoint-reroot (reroot the tree at its midpoint after loading)
- -o <Gingr output>
- -S <output for multi-fasta SNPs>
- -u 0/1 (update the branch values to reflect genome length)
- -v <VCF input>
- -V <VCF output>
- -x <xmfa alignment file>
- -X <output xmfa alignment file>
- -h (show this help)
- -q (quiet mode)

3.1.3 Primary output files

1. Compressed binary archive (GGR)

fixme

3.2 Installation from source

3.2.1 Dependencies:

- Autoconf (http://www.gnu.org/software/autoconf/)
- Protocol Buffers (https://code.google.com/p/protobuf/)
- Zlib (http://www.zlib.net/)

3.2.2 Build

For your convenience, precompiled binaries provides at:

```
http://github.com/marbl/harvest
```

otherwise, to install from source:

```
 git clone https://github.com/marbl/harvest-tools.git hvt_src
 cd hvt_src

./bootsrap.sh
./configure [--prefix=...] [--with-protobuf=...]
```

(continues on next page)

(continued from previous page)

```
make
[sudo] make install
```

3.2.3 Products:

- command line tool (<prefix>/bin/harvest)
- static library (<prefix>/lib/libharvest.a)
- **includes**
  - Harvest, Protocol Buffer message class (<prefix>/include/harvest.pb.h)
  - HarvestIO, Harvest wrapper with file IO (<prefix>/include/HarvestIO.h)

3.2.4 Additional notes:

When running ./configure, use –prefix to install somewhere other than /usr/local. Use –with-protobuf if the Protocol Buffer libraries are not in their default location (/usr/local). Zlib should be installed in a standard system location. Sudo will be necessary for 'make install' if write permission

fixme

fixme

fixme

fixme

Resources

All releases ~ Source code ~ Report an issue