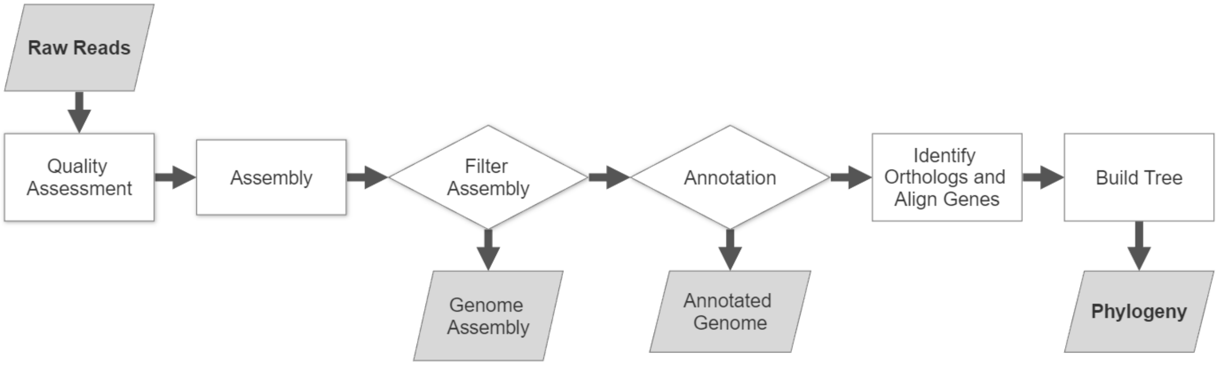
QUICK REFERENCE: Genome assembly, annotation, and building phylogenies from short read data

Compiled by Heidi Abresch, heidi.abresch@umontana.edu

GENERAL OVERVIEW



PROGRAMS OVERVIEW

Program	Documentation	Function
fastQC	https://www.bioinformatics.babraham.ac.uk/projects/fastqc/	Summarizes quality of raw fastq reads.
Trimmomatic	https://github.com/usadellab/Trimmomatic	Trims Illumina paired-end and single-end reads
SPAdes	https://github.com/ablab/spades	Prokaryotic genome and metagenome assembler
Quast	https://quast.sourceforge.net/docs/manual.html	Summary statistics for a set of fasta sequences (usually for a genome assembly)

BLAST (web-based application)	https://blast.ncbi.nlm.nih.gov/Blast.cgi	BLAST identifies regions of similarity between DNA and/or amino acid sequences and compares them to sequence databases and calculates the statistical significance of these similarities.
Local BLAST	https://www.ncbi.nlm.nih.gov/books/NBK279684/	Local BLAST functions similar to web-based BLAST except you can specify specific sequences to compare against.
RAST (web-based application)	https://rast.nmpdr.org/	Used on web browser, not a downloadable program.
Orthofinder	https://github.com/davidemms/OrthoFinder	Identifies orthogroups from group of (related) organisms using amino acid sequences, including single copy orthologs. These are also aligned to each other and output files can be used for downstream applications such as tree building.
IQtree	http://www.iqtree.org/doc/	Assemble a species tree or gene trees from aligned nucleotide or amino acid sequences.

STEP 1: QUALITY ASSESSMENT OF RAW DATA

The first step after getting raw sequence data is to look at the quality of the reads and, if needed, trim the reads of any low-quality sequences. To check the quality of sequences, we use the program fastQC, which analyzes fastq files and outputs a report as an HTML file that summarizes read quality and other important information about the raw data. If the quality is not good or Illumina adapters are still present, we then put the raw reads through the program Trimmomatic which takes raw Illumina reads and can trim any low-quality sequences, based on parameters set by the user, and also remove remaining Illumina adapters.

FastQC

Overview

The fastqc report provides pass/warn/fail checks along with a summary for all of the following categories. Reports are generated for both forward and reverse raw reads.

- Basic Statistics
- Per base sequence quality
- Per tile sequence quality
- Per sequence quality scores

- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented Sequences
- Adapter Content
- Kmer Content

Quick Check Items

For paired end reads, double-check that the number of reads for both forward and reverse files are the same.

For SB metagenome sequencing, a successful sequencing run typically has two peaks of GC content. One in the low 40% and one above 50%.

Using FastQC

```
> fastqc *.fastq.gz -o ./fastqc/
```

Code breakdown:

`<path to raw reads>` REQUIRED. Can be multiple files (usually fwd (R1) and rvs (R2)) Just separate each file with a space. Or point to directory with all files to be QC'd and do *.fastq.gz (this indicates anything that ends with ".fastq.gz"). You can also use regular *.fastq files that are not zipped, however typically raw data files come zipped and you do not need to unzip them to proceed, saving valuable hard drive space.

`-o <path to output directory>` OPTIONAL. Directory must already be created, program will not create output directory. If not provided, outputs will be saved in the current working directory.

If the raw reads have good quality scores, the trimming is not necessary if using for assembly with SPAdes downstream. If adapters still need to be removed from raw sequences, trimmomatic will still be needed even if the quality scores are acceptable.

Trimmomatic

Trimmomatic is a program used to execute various trimming tasks on Illumina paired-end or single-end reads. It is particularly useful in removing or trimming low quality reads. It can also remove sequence adapters if they are still present in the raw reads.

Using Trimmomatic

```
> trimmomatic PE -phred33 -trimlog trimlog.txt <path to
  rawsequence_R1.fastq> <path to rawsequence_R2.fastq> <output path to
  forward paired sequences (sequence_1P.fastq)> <output path to forward
  unpaired sequences (sequence_1U.fastq)> <output path to reverse paired
```

```
sequences (sequence_2P.fastq)> <output path to reverse unpaired
sequences (sequence_2U.fastq)> SLIDINGWINDOW:8:20 MINLEN:20
```

Code breakdown:

`trimmomatic PE`
Indicates that you are entering paired-end sequences. (Alternative is trimmomatic SE)

`-phred33`
Indicates scoring method used on raw reads

`-trimlog <path to trimlog file.txt>`
What to call the file that logs trim steps and where to save it.

`<path to rawsequence_R1.fastq>`
REQUIRED. Path to forward raw reads. Will take .fastq.gz files.

`<path to rawsequence_R2.fastq>`
REQUIRED. Path to reverse raw reads. Will take .fastq.gz files.

`<output path to foward paired sequences (seq_1P.fastq)>`
REQUIRED. Path to forward reads with a retained reverse mate.

`<output path to fowared unpaired sequences (seq_1U.fastq)>`
REQUIRED. Path to forward reads where the reverse mate was removed due to low quality.

`<output path to reverse paired sequences (seq_2P.fastq)>`
REQUIRED. Path to reverse reads with a retained forward mate.

`<output path to reverse unpaired sequences (seq_2U.fastq)>`
REQUIRED. Path to reverse reads where the forward mate was removed due to low quality.

`ILLUMINACLIP:<path to adapter sequence file>:2:30:10`
OPTIONAL. Used only if your raw reads still have adapter sequences

`SLIDINGWINDOW:8:20`
REQUIRED. Scan the read with an 8-base wide sliding window, cutting when the average quality per base drops below 20.

`MINLEN:20`
REQUIRED. Minimum length a read has to be to keep it.

After running trimmomatic, run fastqc again on the trimmed reads to check that the program worked the way you want and to look for any errors. For example, for paired end reads, there should be the same number of reads in the forward and reverse paired sequence files.

STEP 2: GENOME ASSEMBLY

SPAdes

SPAdes (St. Petersburg genome assembler) is a genome assembly program intended for smaller genomes. It can use short-read data (Illumina or IonTorrent reads; paired and unpaired) and can also create hybrid assemblies using a combination of short-read and long-read data from PacBio, Oxford Nanopore and Sanger sequencing. It uses multiple modules including read correction tools for short read data and automatic k-mer selection. Note that as of June 2023, hybrid assembly for metagenomes remains an experimental pipeline and optimal performance is not guaranteed. Additionally, the documentation for SPAdes is extremely thorough and well explained for different types of datasets and uses.

Using SPAdes

```
> spades.py -1 <path to sequence_1P.fastq> -2 <path to
  sequence_2P.fastq> --meta -o <path to output directory>
```

Code breakdown:

`-1 <forward paired reads.fastq.gz>` Provides forward reads.

`-2 <reverse paired reads.fastq.gz>` Provides reverse reads.

`--meta` Indicates sequences are from multiple organisms/genomes and should be assembled as a metagenome. Synonymous with metaspades.py. This is mutually exclusive with --careful option and you cannot provide coverage cutoff values.

`-o <path to output directory>` REQUIRED. Tells program where to output files.

Additional useful options:

`-s <unpaired reads.fastq.gz>` Provides unpaired short reads.

`--isolate` Recommended for high coverage isolate and multicell Illumina data. Not compatible with --careful option.

`--continue`	Continues SPAdes run from the specified output folder starting from the last available checkpoint. If used, the only allowed option is -o.

SPAdes Output Files

SPAdes outputs many files within multiple directories (see https://github.com/ablab/spades#spadesoutput for a detailed list). In most instances, you will only use a few of these files.

- `<output_dir>/scaffolds.fasta`
- `<output_dir>/contigs.fasta`
- `<output_dir>/contigs.paths`
- `<output_dir>/scaffolds.paths`
- `spades.log`

Contigs/scaffolds names in SPAdes output FASTA files have the following format: `>NODE_3_length_237403_cov_243.207`. Where 3 is the number of the contig/scaffold, 237,403 is the sequence length in nucleotides and 243.207 is the kmer coverage for the last (largest) k value used. Note that the kmer coverage is always lower than the read (per-base) coverage.

STEP 3: FILTERING ASSEMBLY AND QUALITY ASSESSMENT

Filtering Assembly using grep and local BLAST

After assembling, we will use the output file scaffolds.fasta and remove any reads that are below 1x coverage or below 1,000 bp in length. These are both somewhat arbitrarily chosen stats, but generally this will remove any assembled pieces that are remaining artifacts from sequencing because SPAdes does not remove these on its own when assembling metagenomes. After this, we can identify the scaffolds that belong to the genome(s) of interest.

Filter out reads above 1x coverage and above 1,000 bp in length:

1. Retrieve sequence IDs for any reads that are above 1x coverage and above 1,000 bp in length.

```
> egrep -o "NODE\_[[:digit:]]{1,}\_length\_[1-
  9][[:digit:]]{3,}\_cov\_[1-9][[:digit:]]{0,}\.[[:digit:]]{1,}"
  scaffolds.fasta > seqIDlist.txt
```

Code breakdown:

`egrep`	Use the extended version of grep
`-o`	Flag to only grab the matched pattern described in quotes
`NODE\_[[:digit:]]{1,}`	grabs the NODE number
`\_length\_[1-9][[:digit:]]{3,}`	grabs the underscore between the NODE number and the length number. Then searches for a length term that is any

	number that starts with a 1-9, and then has 3 or more numbers that follow it
`\_cov\_[1-9][[:digit:]]{0,}\.[[:digit:]]{1,}`	Grabs the underscore between the length number and the coverage number. Then searches for a coverage term that is any coverage number that starts with 1-9 and is followed by any digit 0 or more times, and who has any decimal value.
`inputfile.fasta`	File to search within
`> [outputfile.txt]`	Carat and then the file you want the results to be put into.

2. Make local blast database of scaffolds.fasta (local BLAST covered in detail in next section)

```
> makeblastdb -in scaffolds.fasta -parse_seqids -dbtype nucl
```

3. Retrieve all sequences identified in step 1.

```
> blastdbcmd -entry_batch seqIDlist.txt -db scaffolds.fasta -target_only
  -out scaf_filtered.fasta
```

Quast

QUAST (or Quality Assessment Tool) evaluates and summarizes genome assemblies with a variety of statistics. NOTE: This program can be used on any set of fasta sequences within a single file. Make sure to interpret results accordingly.

Using Quast

```
> quast.py scaf_filtered.fasta -o ./quast/
```

Code Breakdown

Identifying Sequences of Interest Using BLAST

Next, we will use local BLAST to identify scaffolds of interest and then manually refine this list using BLAST online until we only have the set of sequences that belong to the genome(s) we want. Local BLAST works very similar to BLAST on NCBI's website but using the command line. It is particularly useful in that you can blast to only desired sequences, including those not on NCBI's database, and control the output to your heart's content.

STEP 1: Using local BLAST to identify putative sequences

1. Make local blast directories of desired sequences.

```
> makeblastdb -in RgibSB.fasta -parse_seqids -dbtype nucl
```

Code breakdown:

`-in <path to fasta file>` REQUIRED. Tells program which fasta file you'd like to make into a database. Can have multiple sequences in it.

`-parse_seqids` Parses sequence IDs in FASTA format that are delimited using "|" (bar) – e.g. lcl|12345 or gb|54321 – see https://ncbi.github.io/cxx-toolkit/pages/ch_demo#ch_demo.T5 for more information on FASTA sequence ID format

`-dbtype <nucl or protein>` REQUIRED. Tells program whether you are building a database of nucleotide sequences or protein/amino acid sequences.

2. Blast filtered scaffolds to each.

```
> blastn -db RgibSB.fasta -query scaf_filtered.fasta -out blast_SB.txt -
  outfmt 6 -max_hsps 1
```

Local BLAST Output Columns Overview

- Query sequence ID
- Subject sequence ID (Database sequence ID)
- Percent identity
- Length of alignment
- Number of mismatches
- Number of gap openings
- Query start of alignment
- Query end of alignment
- Subject start of alignment
- Subject end of alignment
- E-value
- Bit score (total score)

STEP 2: Manually refine and identify correct scaffolds.

Look at the output files from the local blasts to identify final putative scaffolds for the genome of interest. Use information provided in the blast results such as e-value and bit score (last 2 columns) and compare coverage of different hits. Use BLAST on NCBI's website to confirm or double check any sequences you are not absolutely positive about.

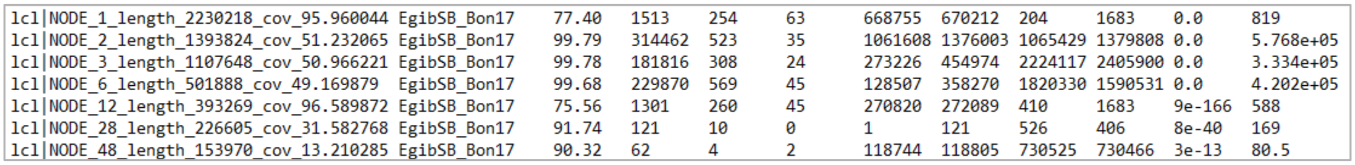
Tips for manually filtering scaffolds:

- The lower the e-value the better!
- The score should be relatively high; longer scaffolds should have higher scores as a longer portion of the reference will align with the query (if they are similar)

- The coverage of all scaffolds from the same genome will be the same, or possibly approx. an equal multiple of the lowest coverage for a duplicated region that has been assembled into 1 region.
- For *Rhopalodia* and *Epithemia* species - in a relatively complete assembly, total length of the SB scaffolds will be ~3 Mb.
- SBs often have an associated plasmid sequence that is ~5,500 bp long and will not necessarily match coverage (or be a multiple) of the chromosome sequences. It is often about 1.5-2x, but not always.

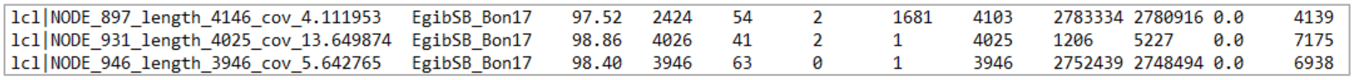
Example:

Below is the first 7 hits for an SB assembly when blasted against the *R. gibba* 17Bon1 SB genome. Node 1 has an e-value of 0 (remember smaller = better for e-value!), but the score (819) is very low compared to the length of the node (2,230,218 bp). Nodes 12, 28, 48 all have high e-values AND low scores. Additionally, nodes with high scores all have about 50x coverage whereas the other nodes have significantly lower or higher coverage values. ***Based on this, we would keep nodes 2, 3, and 6 and remove the others.***



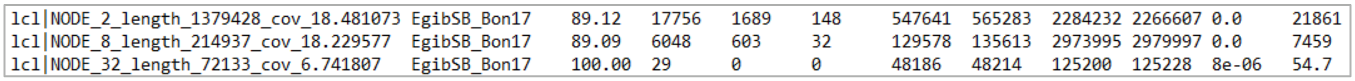
```
lcl|NODE_1_length_2230218_cov_95.960044 EgibSB_Bon17    77.40   1513    254     63      668755  670212  204     1683    0.0     819
lcl|NODE_2_length_1393824_cov_51.232065 EgibSB_Bon17    99.79   314462  523     35      1061608 1376003 1065429 1379808 0.0     5.768e+05
lcl|NODE_3_length_1107648_cov_50.966221 EgibSB_Bon17    99.78   181816  308     24      273226  454974  2224117 2405900 0.0     3.334e+05
lcl|NODE_6_length_501888_cov_49.169879  EgibSB_Bon17    99.68   229870  569     45      128507  358270  1820330 1590531 0.0     4.202e+05
lcl|NODE_12_length_393269_cov_96.589872 EgibSB_Bon17    75.56   1301    260     45      270820  272089  410     1683    9e-166  588
lcl|NODE_28_length_226605_cov_31.582768 EgibSB_Bon17    91.74   121     10      0       1       121     526     406     8e-40   169
lcl|NODE_48_length_153970_cov_13.210285 EgibSB_Bon17    90.32   62      4       2       118744  118805  730525  730466  3e-13   80.5
```

The example below is from a different assembly where most scaffolds belonging to the SB are about 4-6x coverage. Node 931 has much higher coverage, but the percent identity, e-value, and total score are all good values. If you BLAST this on NCBI's website, you'll find that this is in fact an SB sequence. The higher coverage indicates that it is probably a region that is duplicated in the genome but has been misassembled into a single scaffold. These sequences with high coverage should always be approximately whole number multiple of the rest of the scaffolds for the genome.



```
lcl|NODE_897_length_4146_cov_4.111953    EgibSB_Bon17    97.52   2424    54      2       1681    4103    2783334 2780916 0.0     4139
lcl|NODE_931_length_4025_cov_13.649874   EgibSB_Bon17    98.86   4026    41      2       1       4025    1206    5227    0.0     7175
lcl|NODE_946_length_3946_cov_5.642765    EgibSB_Bon17    98.40   3946    63      0       1       3946    2752439 2748494 0.0     6938
```

In the final below example, node 32 has 100% identity, but a really high e-value and low score. This pattern of high percent identity but high e-value/low score is common in longer assembled sequences that include highly conserved regions such as ribosome sequences.



```
lcl|NODE_2_length_1379428_cov_18.481073 EgibSB_Bon17    89.12   17756   1689    148     547641  565283  2284232 2266607 0.0     21861
lcl|NODE_8_length_214937_cov_18.229577  EgibSB_Bon17    89.09   6048    603     32      129578  135613  2973995 2979997 0.0     7459
lcl|NODE_32_length_72133_cov_6.741807   EgibSB_Bon17    100.00  29      0       0       48186   48214   125200  125228  8e-06   54.7
```

If you're still not sure based on any of these statistics, you can blast these sequences using BLAST on NCBI's website. If there are a lot of questionable sequences, you can also use GC content of each scaffold to help filter before manually BLASTing any sequences.

STEP 3: Generate fasta files with only putative scaffolds for mt, cp, SB.

1. Cut the blast results down to a list of seqIDs in a single column.

TIP: If there are a lot of sequences to cut, you can instead do this in the command line by saving the BLAST output with only the rows that have the correct sequences and generate the seqIDs as follows:

```
> cut -f 1 <file_to_edit.txt> > <seqids.txt>
```

This code simply cuts out the first column of a tab delimited text file and saves it to a new file.

2. Save that file and put it into your working directory and collect those sequences using blastdbcmd. Here, I just pull sequences from the original scaffolds.fasta file using the database we created when we filtered the original assembly. I do not need to create a 2nd BLAST database out of the filtered scaffolds because they both contain the sequences of interest with the same sequence IDs.

   ```
   > blastdbcmd -entry_batch seqIDs.txt -db scaffolds.fasta -target_only -
     out SBscafs.fasta
   ```

Code breakdown:

`-entry_batch <path to seqID file.txt>`	REQUIRED.
`-db <path to file to search.fasta>`	REQUIRED. Path to fasta file to search for sequence IDs provided in -entry_batch
`-target_only`	Tells program to only retrieve sequences that match those listed in the -entry_batch file
`-out <output_file.fasta>`	REQUIRED. Tells program where to save the identified sequences.

STEP 4: GENOME ANNOTATION

RAST

Rapid Annotation using Subsystem Technology, or RAST, is an online prokaryotic genome annotation service. It can be used for bacterial annotations as well as mitochondria and chloroplast annotations. While RAST is not the most thorough or accurate annotation service, it is fast and provides a good first pass of a genome. (For more thorough annotations, I recommend using NCBI's Prokaryotic Genome Annotation Pipeline). To use RAST, you need to have an account.

The benefit of using RAST is the user interface within the browser called The SEED Viewer. Within the web browser, you can view stats about the sequence(s) uploaded and summaries of the annotation (e.g. number of RNAs, coding sequences, subsystem content). When browsing the genome, you can also search for functions and subsystems within the genome browser and see a graphic of features within their coding frame in a region of the genome.

Using RAST

1. Upload sequences to RAST:
   a. On RAST, go to "Your jobs" -> Upload new job.

b. Upload the sequences file you sorted in the previous step (e.g. SBscafs.fasta) and go to step 2.
c. Fill in desired taxonomy information and select translation code 11.
d. Go to step 3 and do not change any of this information unless otherwise specified.
e. Wait for the job to finish! You can check your status under Your Jobs -> Job overview

2. Once the job is finished, you can browse the annotated genome within your browser using the built in viewer. You can also filter and search within the annotated genome here. RAST also provides many different useful file types based on the annotation including .gff3, .faa, and .gbk files.
3. For step 5, we will need the .faa file. You can download this file from the Job Details page under Available downloads for this job -> Amino-acid fasta file

STEP 5: IDENTIFYING AND ALIGNING SINGLE-COPY ORTHOLOGS

Orthofinder

Orthofinder is another program with many uses – primarily it is used to identify orthogroups and align single copy orthologs shared across a group of genomes. It also has extensive documentation that describes different use cases and details options and output files. There are also a growing number of guided walkthroughs (written and video formats) of how to use orthofinder and the different output files.

Using Orthofinder

1. First you will need to compile all amino acid fasta files (.faa) for all genomes that you are interested in comparing. All of these files should be put into a single directory and no other files should be present in this directory.
2. Run orthofinder on files.

```
> orthofinder -f <path to faa directory> - M msa -oa
```

Code breakdown:

`-f <path to faa directory>` Tells program where files are and indicates FASTA format

`-M msa` indicates multiple sequence alignment. This option is only needed if you want to infer maximum likelihood trees from multiple sequence alignments (MSA). Uses MAFFT by default for aligning.

`-oa` stops pipeline after msa step (requires -M msa option)

Orthofinder Output

Similar to SPAdes, there are a lot of output files provided by Orthofinder, and often you will only use a handful of these. The most useful for our purposes are a few of the summary output files and the file that includes a concatenated alignment of all the single copy orthologs.

- `/Statistics_Overall.tsv`
- `/Statistics_PerSpecies.tsv`
- `/Orthogroups_SpeciesOverlaps.tsv`
- `/Orthologues/Alignments/SpeciesTreeAlignment.fa`

Troubleshooting Tips

If you have very few single copy orthologs, it is possible that one of the .faa files is truncated or the species is too different for orthofinder to correctly infer orthogroups. You can look at this by examining the `Statistics_PerSpecies.tsv` file and the `Orthogroups_SpeciesOverlaps.tsv` file. I recommend opening the latter in excel and conditionally formatting the data to highlight values as a gradient from low to high. This provides a quick visual pattern of what taxa the lowest shared values are coming from.

STEP 6: TREE BUILDING

For this, we will be creating a species tree in IQtree using the aligned single copy orthologs from orthofinder. However, IQtree also supports creating gene trees from multiple gene alignment files which can be useful in other scenarios.

First, you will want to move or copy the file `SpeciesTreeAlignment.fa` from the orthofinder output in the Alignments directory to your IQtree working directory.

IQtree

IQtree is a robust program for tree building using nucleotide or amino acid data. They also have a web server that is free to use. This program also has amazing documentation and tutorials on their website and has an active development team. Note: Here we use IQtree version 1, but IQtree version 2 was released in 2020.

Using IQtree

```
> iqtree -s SpeciesTreeAlignment.fa -bb 1000 -alrt 1000 -nt AUTO -m TEST
  -pre speciestree
```

Code Breakdown

To get a species tree from a regular alignment (1 concatenated sequence per species):

`-s <alignment file>`	*ignores blanks (-) and ambiguous nts (N) when creating trees
`-bb 1000`	number of bootstrap tests

`-alrt 1000`	sh-like test
`-nt AUTO`	specifies number of threads
`-m TEST`	tells iqtree to determine the model of genome evolution to use
`-pre`	specify a prefix for all output files

ALTERNATIVE USE: To generate gene trees from multiple gene alignment files:

`-S <input directory>`	*ignores blanks (-) and ambiguous nts (N) when creating trees
`-bb 1000`	see above
`-alrt 1000`	see above
`-nt AUTO`	see above

SUPPLEMENTAL PROGRAMS

This is a list of other useful programs that are commonly used to analyze genome assemblies and compare different genome sequences.

Program	**Documentation**	**Notes**
kraken2	https://ccb.jhu.edu/software/kraken2/	Assigns taxonomy to reads or longer assembled sequences to a database of sequences. Can use provided database as well as add your own sequences to the database to search within.
EukRep	https://github.com/patrickwest/EukRep	Takes metagenomic assembly and assigns sequences as eukaryotic or prokaryotic and can separate these sequences into separate files.
samtools	http://www.htslib.org/	Tool with lots of applications for files in SAM (Sequence Alignment/Map), BAM, and CRAM formats.

Burrows-Wheeler Aligner (bwa)	https://bio-bwa.sourceforge.net/	BWA is a software package for mapping reads or sequences to a reference genome.
Integrative Genomics Viewer (IGV)	https://igv.org/ https://software.broadinstitute.org/software/igv/	The Integrative Genomics Viewer (IGV) is an interactive tool for the visual exploration of genomic data. It supports all the common types of genomic data and metadata, investigator-generated or publicly available, loaded from local or cloud sources.
Mauve	https://darlinglab.org/mauve/mauve.html	Mauve is a prgraom for constructing multiple genome alignments and visualizing synteny in the presence of large-scale evolutionary events such as rearrangement and inversion.
Prokaryotic Genome Annotation Pipeline (PGAP) from NCBI	https://www.ncbi.nlm.nih.gov/genome/annotation_prok/	PGAP annotates bacterial and archaeal genomes that combines *ab initio* gene prediction algorithms with homology based methods. It predicts protein coding genes, structural RNAs, tRNAs, small RNAs, pseudogenes, control regions, direct and inverted repeats, insertion sequences, transposons and other mobile elements.
Roary	https://sanger-pathogens.github.io/Roary/	Takes .gff3 files as input and provides the core and pan genome for files provided.