# Phylogenetic Placement & Taxonomic Identification

with

# SEPP and TIPP (and alignment with PASTA)

---

Michael Nute

STAMPS 2019

Woods Hole, MA

# Agenda

*If you have Windows, this section will be very useful if you want to run much of the software that will be discussed later in this course.*

**SEPP/TIPP:**

- Step-by-step Tutorial: https://github.com/MGNute/stamps-tutorial
  - Includes link to pdf of these slides
- Quick Review & Practical Considerations
  - SEPP vs. TIPP comparison
  - SEPP/TIPP: for Shotgun or Amplicon Data?
  - SEPP: interpreting branch length
- Additional Reference Material
  - TIPP: BLAST pipeline for shotgun data
  - TIPP reference package link & contents
  - Guide to SEPP command-line arguments
  - Guide to TIPP command-line arguments
  - Contents of TIPP/SEPP placement JSON file
  - Additional Resources

**PASTA:**

- Installation Options
- Installing the GUI:
  - Getting Ubuntu for Windows
  - Installing the PASTA GUI on Ubuntu (including for Windows)
  - Installing the PASTA GUI on Mac from the .dmg file
- Quick Review of PASTA Algorithm
- PASTA GUI User Guide
- Final Tips and Best Practices

# Step-by-Step Tutorial:

- **Part I**: Taxonomic Identification using TIPP
  - Test Data: Lemur vaginal swab, 16S amplicon sequencing with 454.
  - Reference Data: 11,988 full-length sequences from RDP
    - *For the tutorial: Clostridia only*
  - **Part I.a**: visualizing the placements from TIPP
- **Part II**: Phylogenetic Placement using SEPP
  - Test Data: 5 reads selected from the data above
  - Reference Data: (*same as above*)
- **Part III**: Abundance Profiling using TIPP
- **Part IV**: Multiple Sequence Alignment using PASTA
  - Test Data: 96 gyrA amino acid sequences from NCBI
- **https://github.com/MGNute/stamps-tutorial**
  - Follow the link for the “Hands-On Worksheet” to get started.
  - Feel free to get started and let us know if you have questions!

# TIPP vs. SEPP

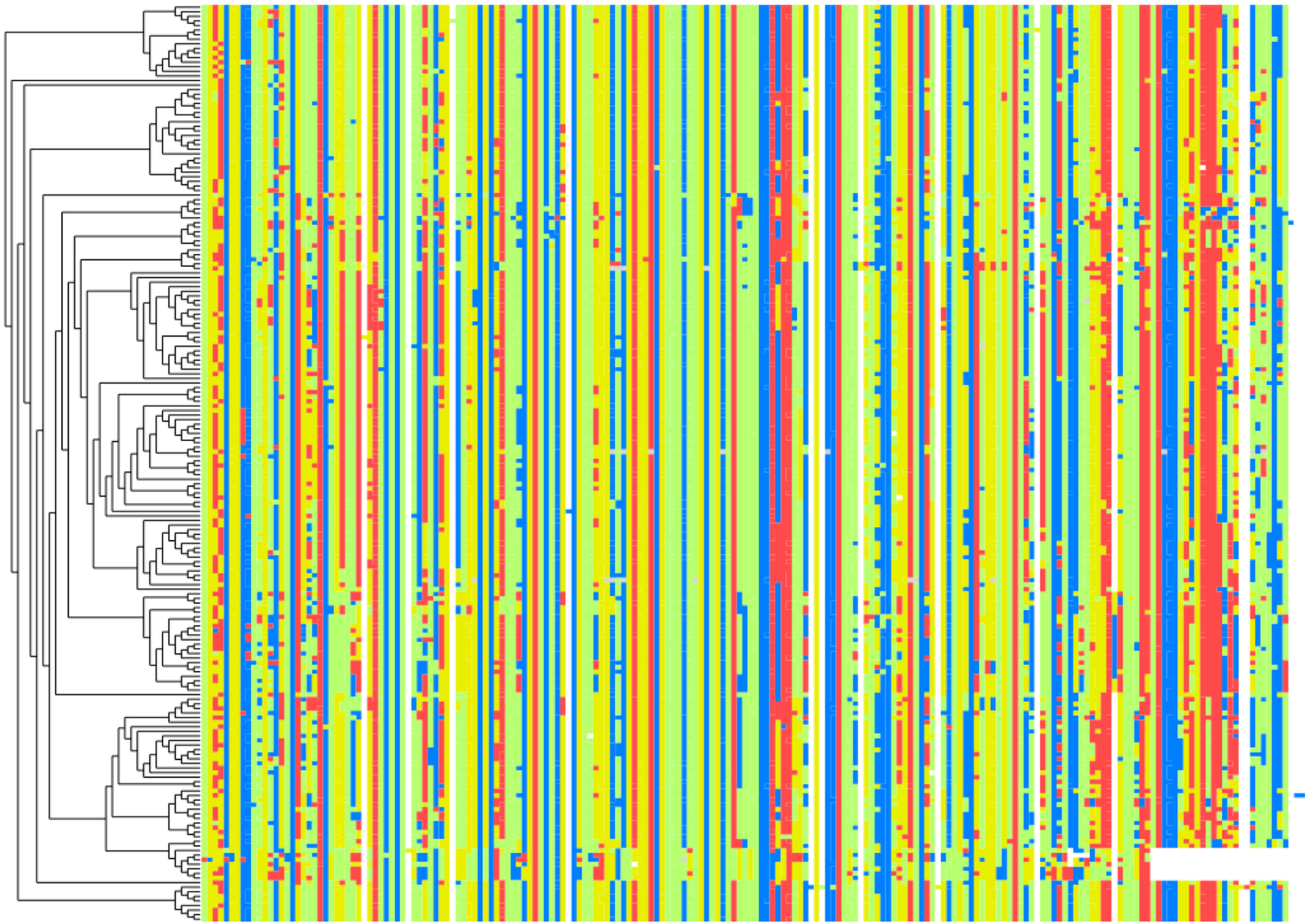
| | SEPP<br>*"Placement in a Phylogeny"* | TIPP<br>*"Placement in a Taxonomy"* |
|---|---|---|
| **Output** | • Attachment point and branch length for each read.<br>*(esoteric)* | • Taxonomic Identification for each read.<br>*(concrete)* |
| **Reference Tree** | • Phylogeny<br>*(based on gene, may not agree with taxonomy)* | • Taxonomy |
| **Used for** | • Downstream analysis, data exploration (visualization), etc… | • Taxonomic Identification<br>• Abundance Profiling |
| **Advantage** | • Placement provides richer data for metrics like Unifrac (Jansenn, et. al, *mSystems*, 2018)<br>• Branch length can be informative | • More accurate than BLAST when novel sequences are present |

# TIPP & SEPP: Shotgun or Amplicon?

*Both!!*

- **Shotgun Sequencing**
  - Reads should be filtered down to subset that hits the reference tree (using BLAST, e.g.)
  - Reads should be corrected to go the same direction as the reference, and trimmed where edges are overlapping (TIPP will do this automatically via run_abundance.py, *see slides 8-9*).
  - For TIPP, can be done for many references separately and the results combined
    - *See run_abundance.py example later (again, slides 8-9)*
  - Reference Packages:
    - 40 COGS usable as marker genes (single-copy, universal) available with full reference packages (*link on slide 10*)
- **Amplicon Sequencing**
  - No need to filter reads (they already come from a single known gene)
  - Can use representative sequence from ASVs (ideal)
  - Reference Packages:
    - In tipp.zip, package “RDP_Bact_2016” is recent and high-quality for 16S amplicon data.
    - For other genes, roll your own reference package (or contact me for help at nute2@Illinois.edu)

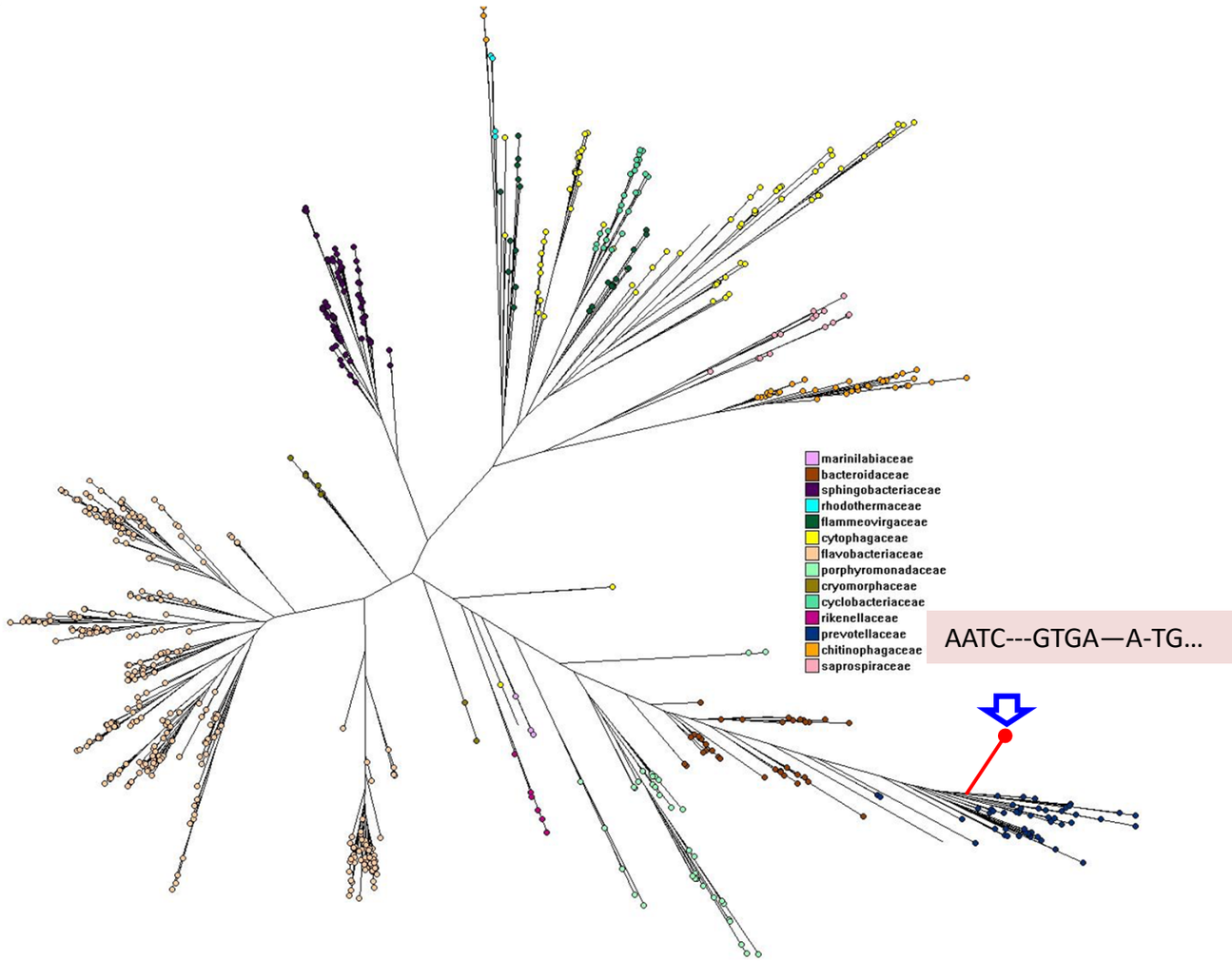
# Reference Package (Alignment & Tree)



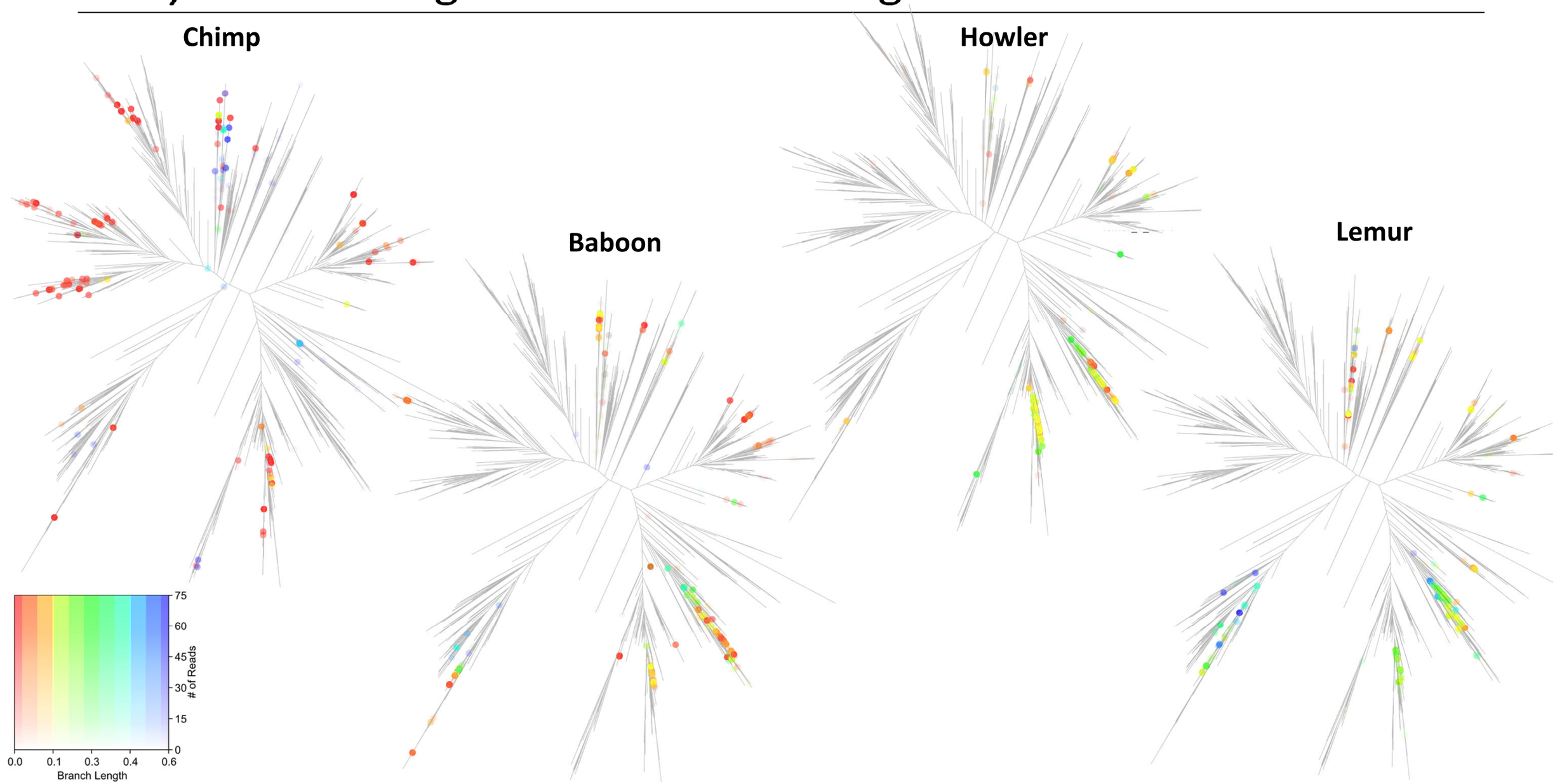
**Full-length Sequences** with quality multiple sequence alignment

# SEPP (or TIPP): Interpreting Branch Length

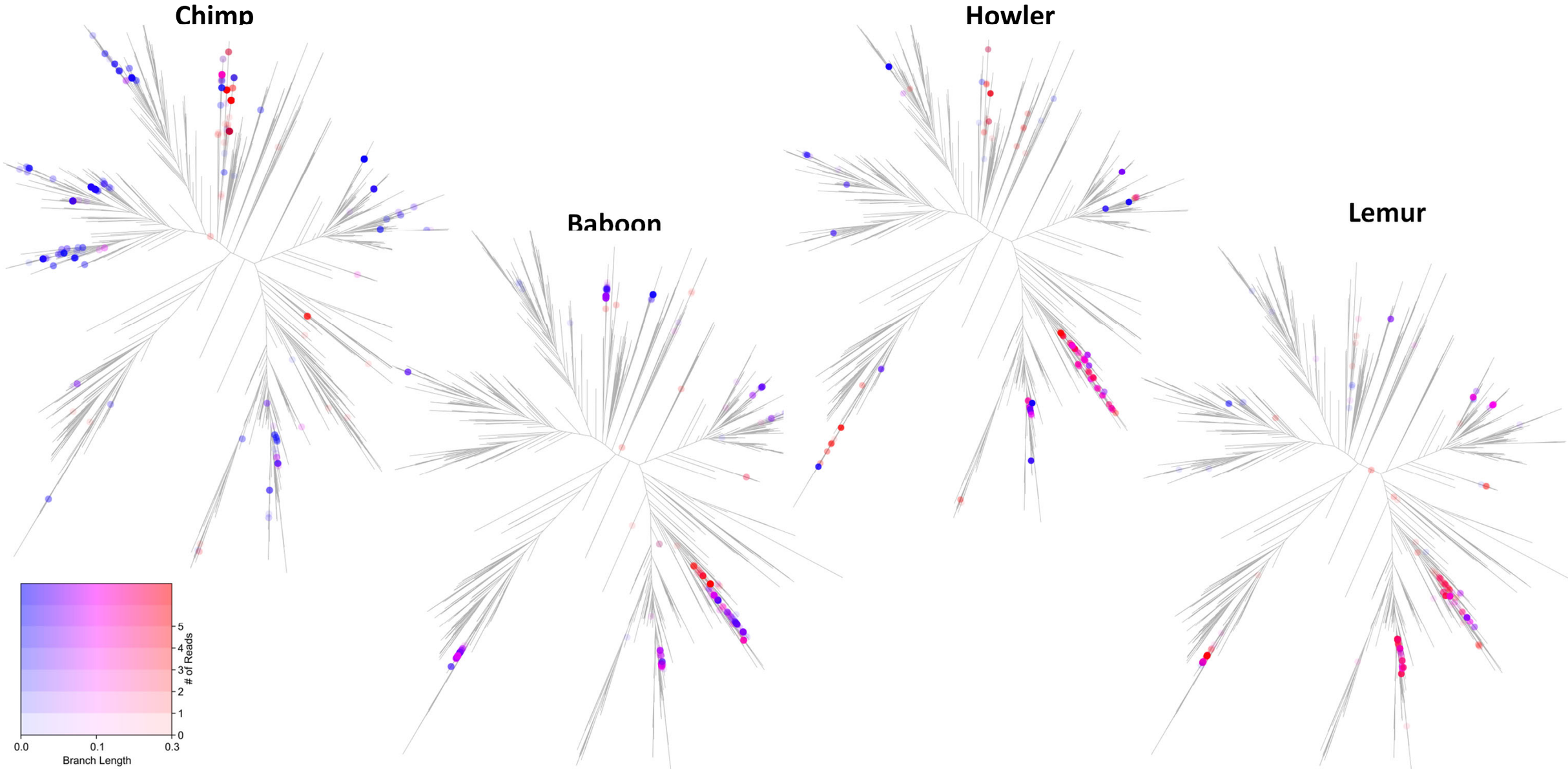
- For each read, SEPP gives the best ***new branch*** in the tree.
- Includes:
  1. Attachment Point
     - *Attaches near the closest matching sequence(s) in the alignment.*
  2. Branch Length
     - *How far it is from its closest match in the database.*
     - *In some sense, a measure of **novelty**.*
  3. Alignment to Reference
     - *For amplicon data, should be as expected*



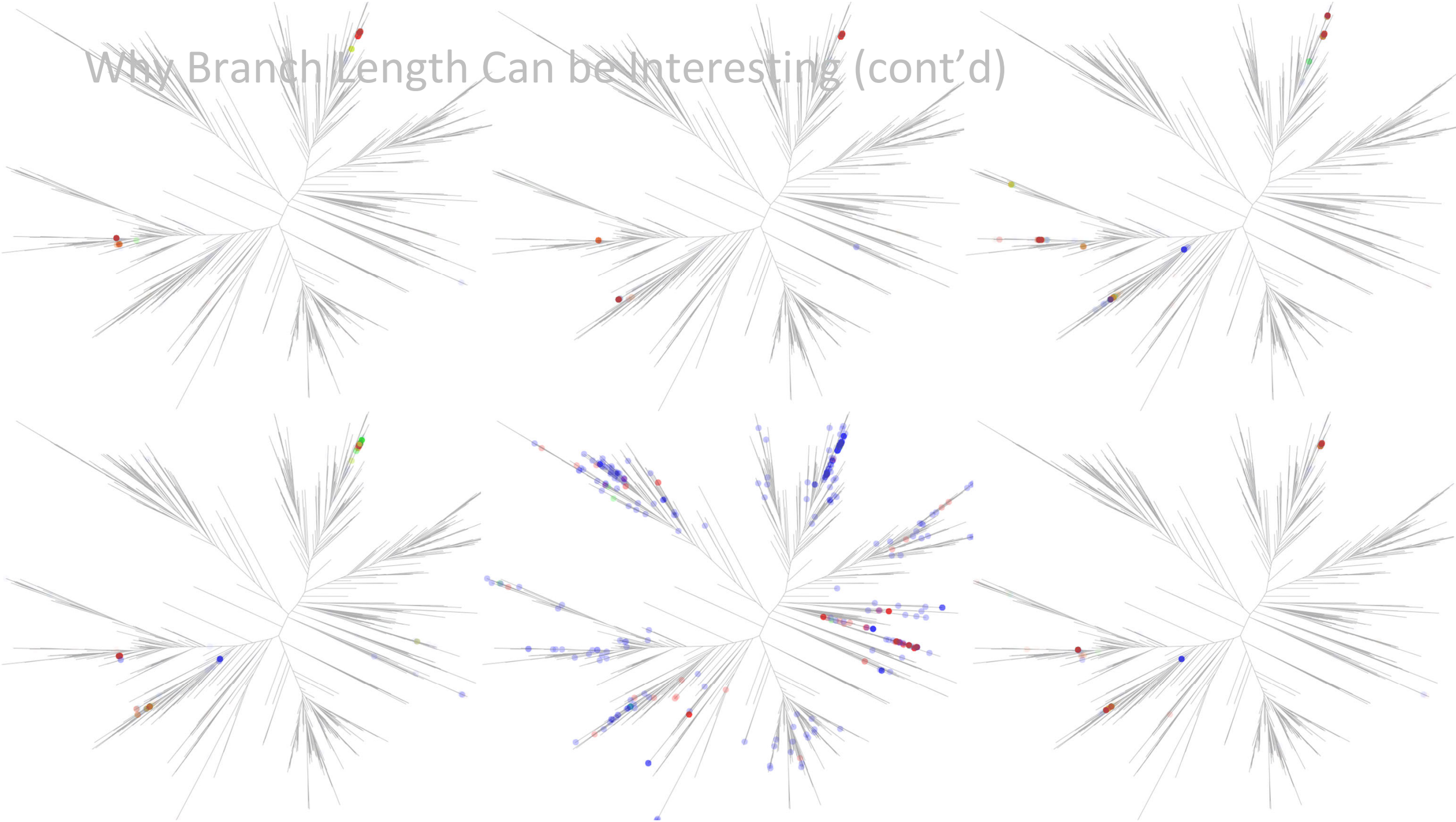
# Why Branch Length Can be Interesting



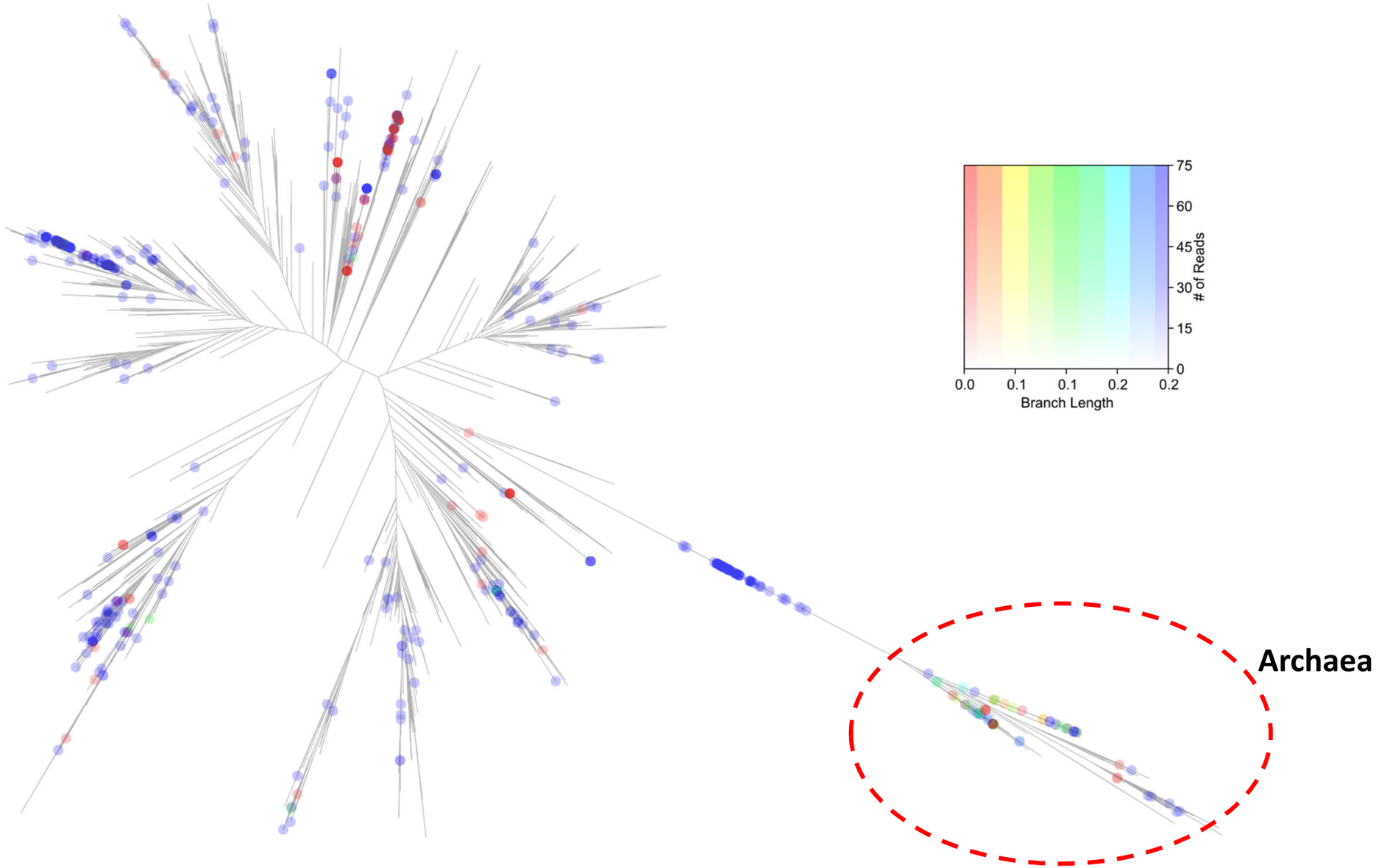
# Why Branch Length Can be Interesting (Accessible)



# Why Branch Length Can be Interesting (cont'd)



# ??


75
60
45
30
15
0
# of Reads
0.0
0.1
0.1
0.2
0.2
Branch Length
Archaea

# TIPP: Pipeline for Shotgun Data

- For future reference (already loaded on STAMPS server):
  - TIPP Reference Package Download: (TBA)
- For shotgun data, use 40 single-copy marker genes as references:
  - First step: use BLAST to filter out reads that hit any of the 40.
  - Second step: use TIPP to place into the taxonomy & classify
  - Third step: summarize
- run_abundance.py script executes the entire pipeline. (NOTE: this will mean up to 40 separate TIPP runs, so it may be best to divide the inputs.)
- Sample command:

```
python3 $sepppath/run_abundance.py \
      -c $sepppath/.sepp/tipp.config \
      -G cogs \
      -A 10 -P 2500 \
      -at 0.00 -pt 0.00 \
      -f <input_seqs> \
      -d <output_folder> -o <job_name> -p <temp_dir>
```

# TIPP: Pipeline for Shotgun Data (continued)

- First output from run_abundance.py is BLAST results.
- To re-start, or re-run on old BLAST results: add ‘-b’ option:

```
python3 $sepppath/run_abundance.py \
      -c $sepppath/.sepp/tipp.config \
      -G cogs \
      -A 10 -P 2500 \
      -at 0.00 -pt 0.00 \
      -f <input_seqs> \
      -d <output_folder> -o <job_name> -p <temp_dir> \
      -b <blast_output_file>
```

# TIPP Reference Package Contents

| File Name | Description |
|---|---|
| **all_taxon.taxonomy** | *(used internally within TIPP)* |
| **original_sequence_name_map.txt** | Sequences in reference data get renamed to avoid issues with special characters. This is the mapping to their original names from NCBI. |
| **pasta.fasta** | Multiple sequence alignment of reference sequences. |
| **pasta.hmm** | *(used internally within TIPP)* |
| **pasta.size** | *(used internally within TIPP)* |
| **pasta.taxonomy** | Refined taxonomy on reference sequences. Conforms to the taxonomy described by "taxonomy.table". |
| **pasta.taxonomy.RAxML_info** | RAxML info file for the refined taxonomy "pasta.taxonomy" |
| **pasta.tree** | Phylogeny estimated on reference sequences. Does not conform to the taxonomy. |
| **pasta.tree.RAxML_info** | RAxML info file for the phylogeny in "pasta.tree" |
| **species.mapping** | Tab delimited file with sequence name and NCBI Taxomy ID for all sequences in the reference alignment. |
| **species.txt** | *(used internally within TIPP)* |
| **taxonomy.table** | Taxonomy structure for taxa covered by reference alignment. |

*Beta versions of updated TIPP reference packages are located at: https://www.dropbox.com/s/98r7r9ccf1zgyqt/tipp-2017.zip*

# SEPP Command Line Tips

*Use command `run_sepp.py –h` to show help and all command line options*

**Categories:**

- Decomposition Options:
  - It is ok to leave these all blank
  - SEPP has sensible defaults
- Input Options:
  - **Mandatory**:
    - Input Tree (-t)
    - Input Alignment (-a)
    - RAxML model file (-r)
    - Query sequences, a.k.a. fragment file (-f)
- Output Options:
  - Good idea to specify all of these:
  - Temp directory (-p)
    - Needs to be cleaned out periodically.
  - Output folder (-d)
    - (Good practice to know where output will go)
  - Output prefix (-o)
    - Also good practice, to avoid confusing one run with another.

# TIPP Command Line Tips:

*Use command `run_tipp.py –h` to show help and all command line options*

**Categories:**

- Decomposition Options:
  - (Same as SEPP, leaving blank is fine)
  - Can affect output and running time though.
- Input Options:
  - **Two Options**:
    - Specify same inputs as SEPP (on previous page)
    - Specify a reference package (-R)
      - Must be in the folder "$REFERENCE" at install time.
- TIPP-specific options:
  - Taxonomy (-tx): called "taxonomy.table" in the reference packages
  - Taxonomy-Name Mapping (-txm): called "species.mapping" in the reference packages
- Another Option: use run_abundance.py to implement the full pipeline on a set of shotgun data.

# Additional Resources

- GitHub repository:
  - https://github.com/smirarab/sepp
  - Maintained by Siavash Mirarab, may not always have latest development (but well-tested and stable)
- TIPP Reference Packages:
  - https://www.dropbox.com/s/98r7r9ccf1zgyqt/tipp-2017.zip
- SEPP Visualization (warning: very preliminary):
  - https://github.com/MGNute/pican_pi
- Contact the TIPP developers:
  - Tandy Warnow (warnow@Illinois.edu)
  - Mike Nute (nute2@Illinois.edu)
  - Siavash Mirarab (smirarabbaygi@eng.ucsd.edu)

# PASTA: Installation

See detailed installation instructions at:

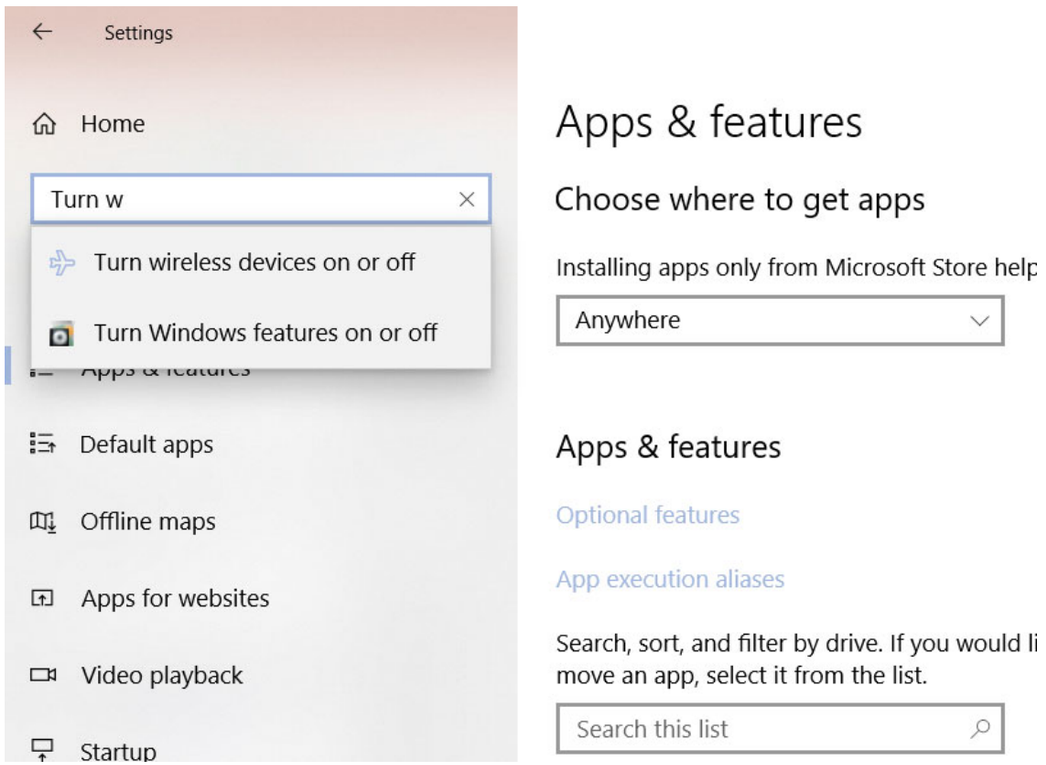
https://github.com/smirarab/pasta

Options:

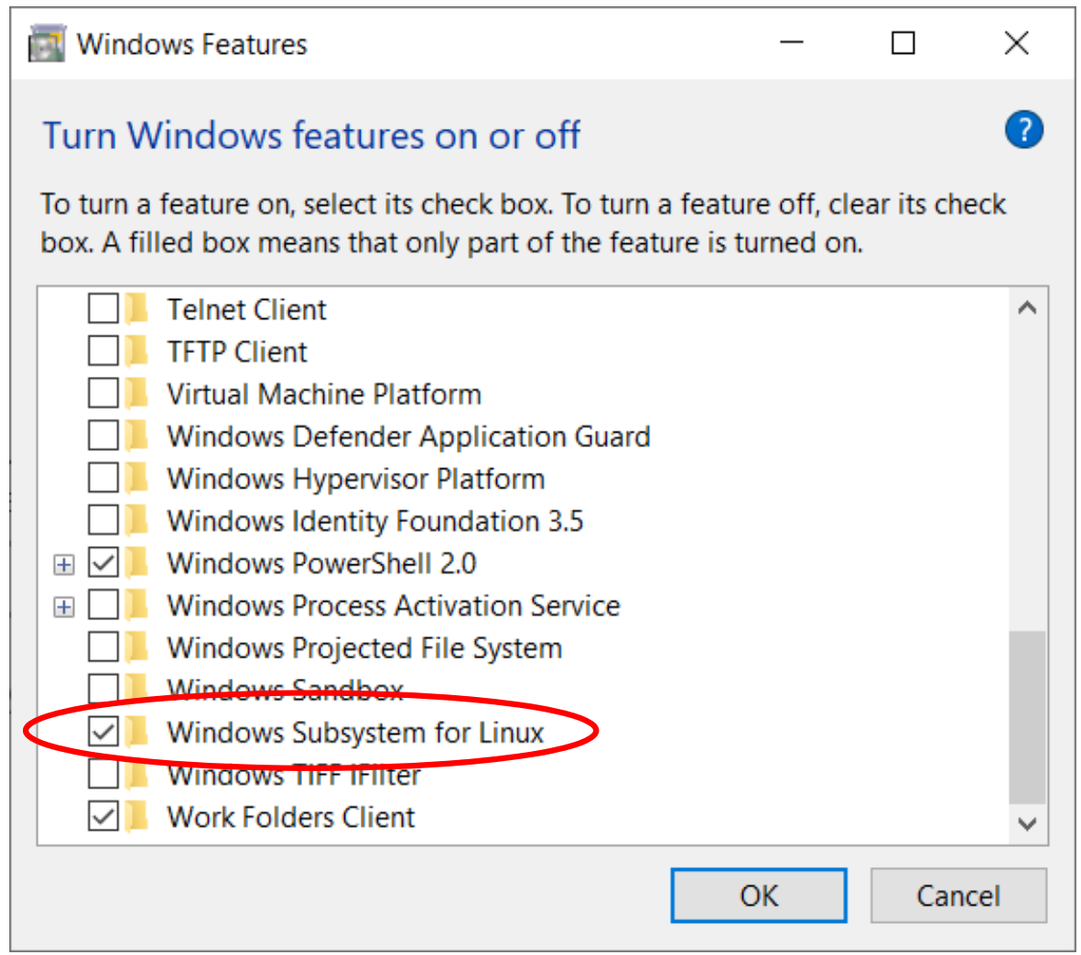
1) MAC
   - DMG file available at the link above (easiest), or Docker file (second easiest)
   - Install from instructions at link above
2) Linux
   - Detailed instructions available at the link above
   - Requires JAVA, wxPython*
3) Windows
   - Must find a Linux option: either Ubuntu app in Microsoft Store (see next slide) or VM using VirtualBox

# Getting Ubuntu on Windows (1 of 3)

1. Open Settings
2. Search for “Turn Windows Features on or off”

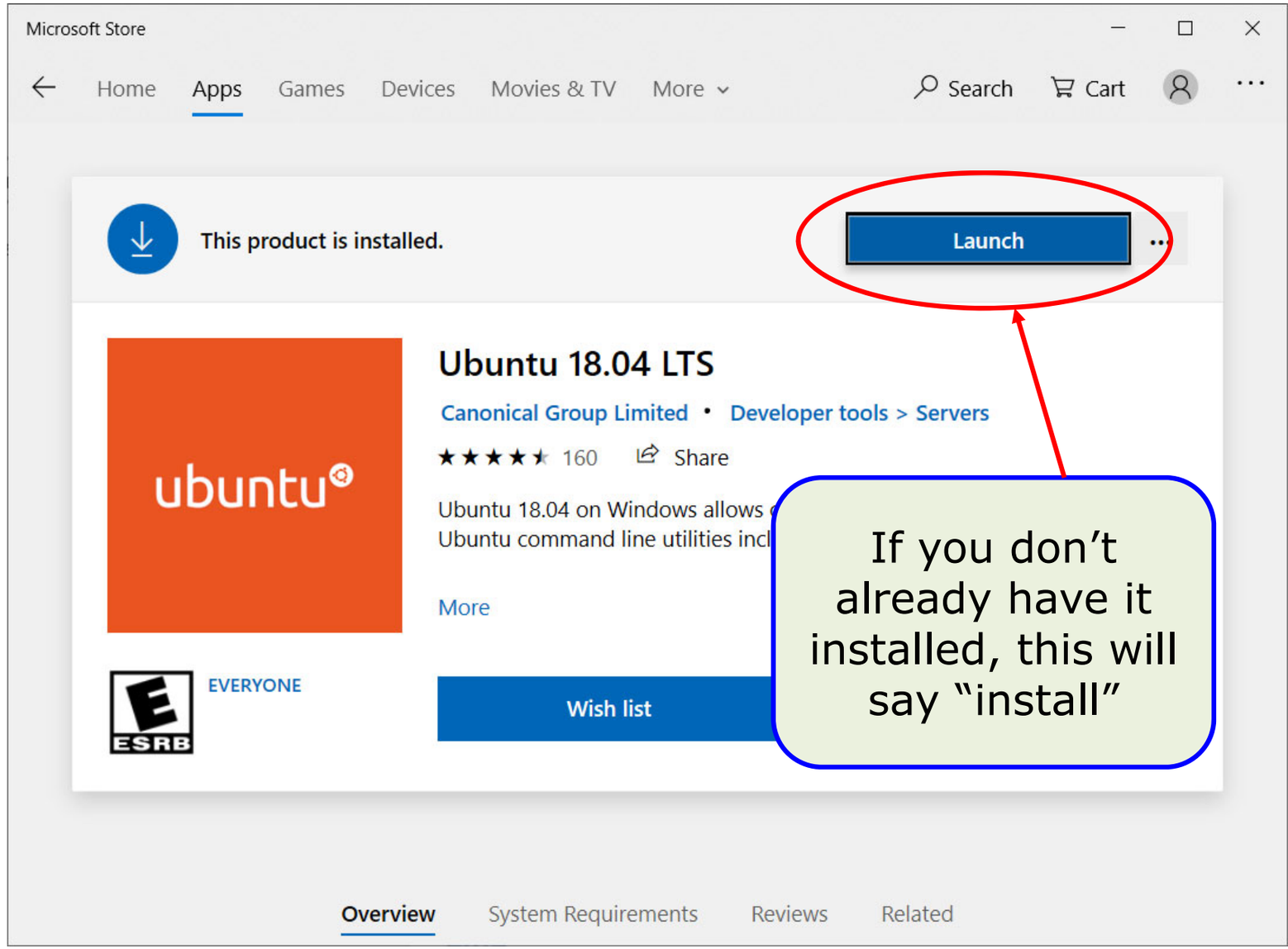



3. In the ensuing dialoge, check “Windows Subsystem for Linux”



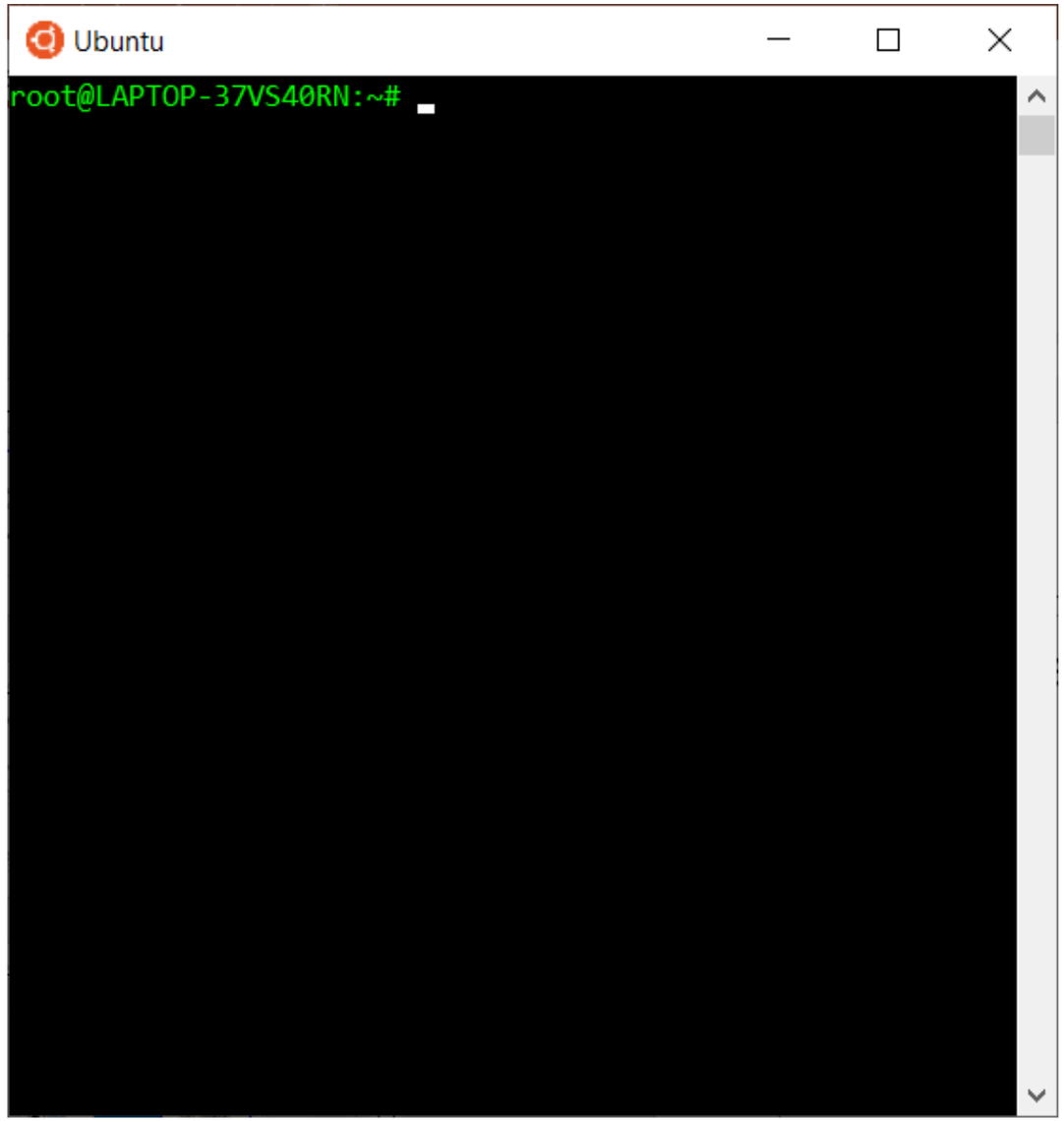
# Getting Ubuntu on Windows (2 of 3)

4. Open the Microsoft Store and Search for “Ubuntu”
5. Install “Ubunutu 18.04 LTS” or whatever the latest version is.



# Getting Ubuntu on Windows (3 of 3)

6. Launch the app and follow the prompts at startup
7. Run "`sudo apt-get update`" to check that everything went well and make sure the package manager is up-to-date.

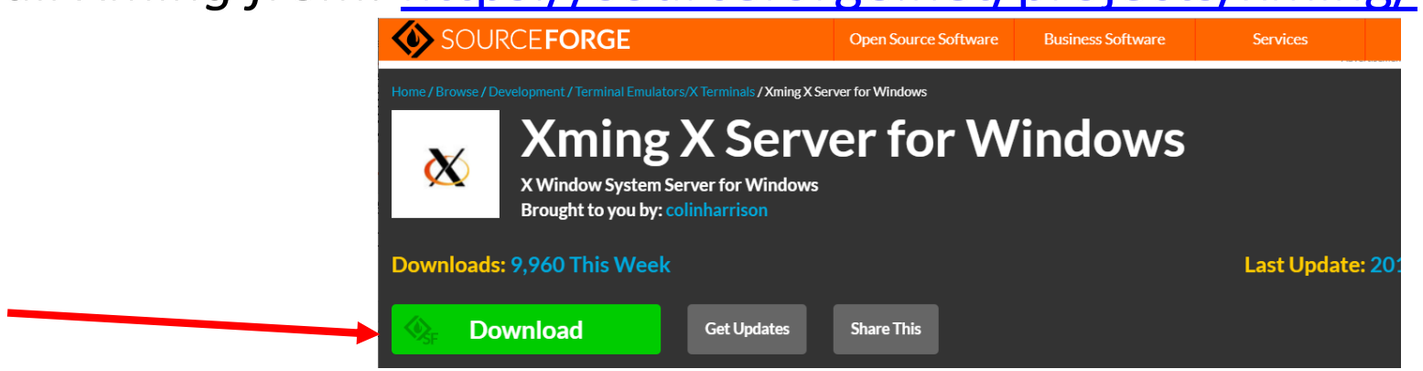


# Installing wxPython on Linux (including Ubuntu on Windows)

*Command (description):*

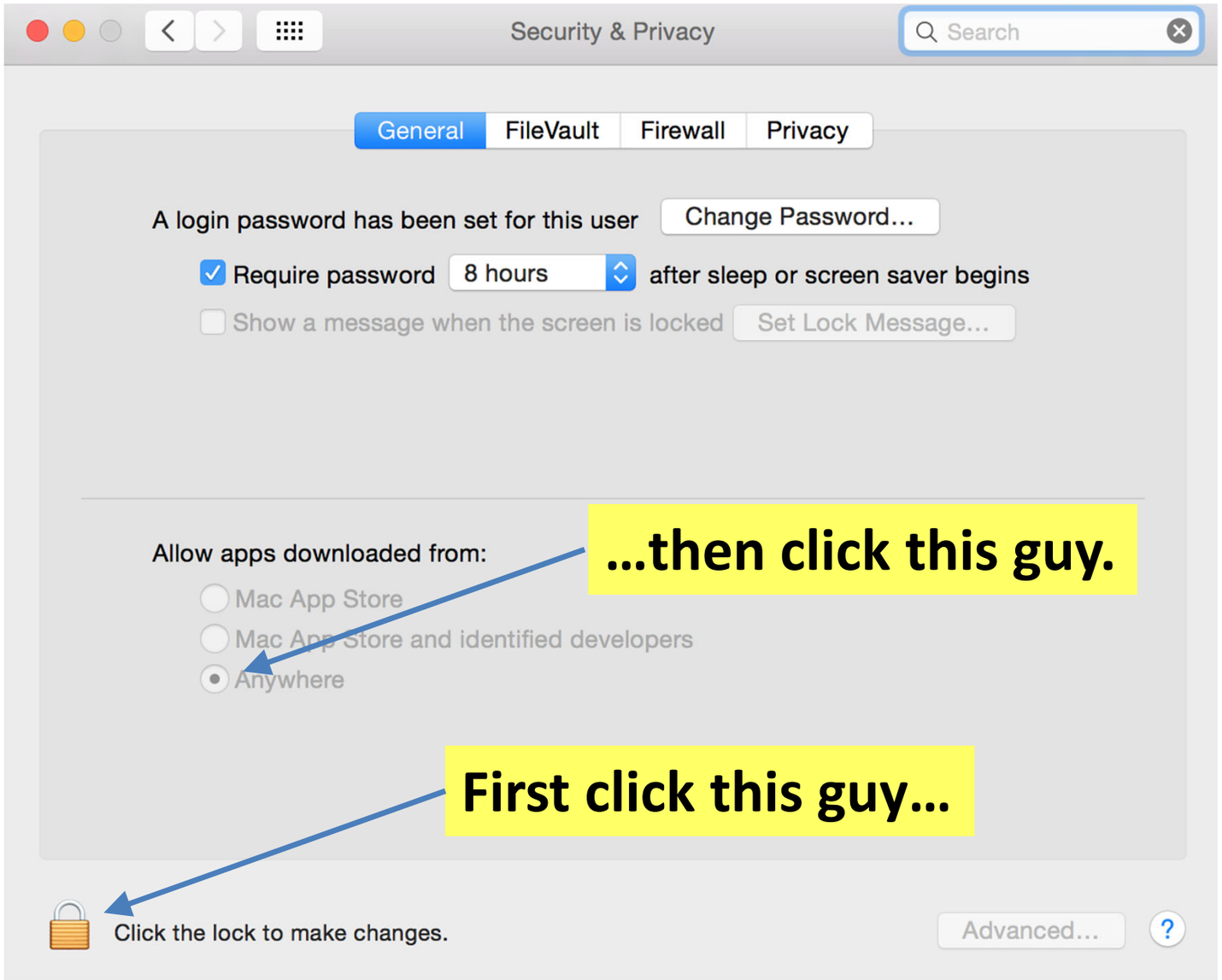
| | |
|---|---|
| **`sudo apt-get update`** | *if you haven't already* |
| **`sudo apt install python3-pip`** | *install PIP, the python package installer* |
| **`sudo apt install libgtk-3-dev`** | *installs GTK* |
| **`sudo apt install default-jdk`** | *installs Java, which PASTA uses* |
| **`pip3 install pathlib2`** | *PIP should install this with wxPython, but there is a bug in that package currently* |
| **`pip3 install wxPython`** | *the package that the GUI runs in* |

*Finally: on Windows, install Xming from:* https://sourceforge.net/projects/xming/



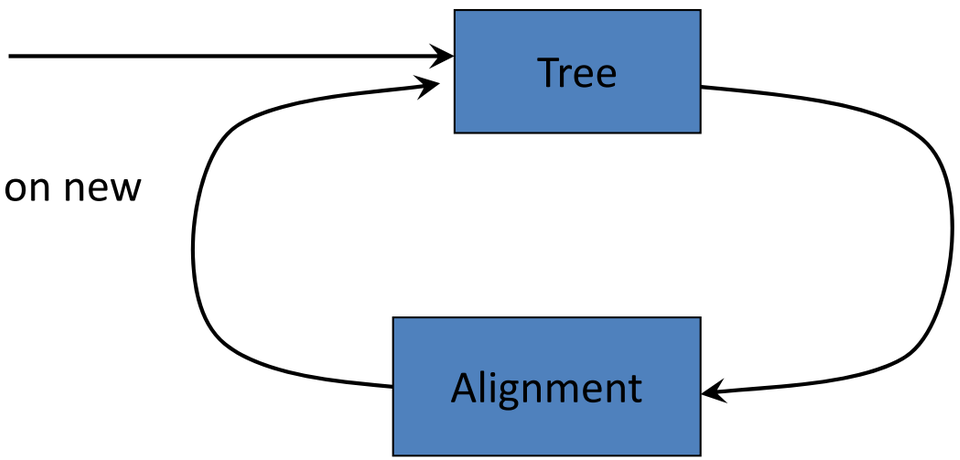
# PASTA: Installing with a Mac

- If you have a mac, download the .dmg file (link is on the github page).
- Open the dmg and copy the app to a folder you want to use.
- You may need to update settings to allow you to run software downloaded from the internet. Go to **System Preferences → Security & Privacy** and you will see the screen on the right:



# SATé and PASTA Algorithms (from Tandy's talk)

Obtain initial alignment and estimated ML tree



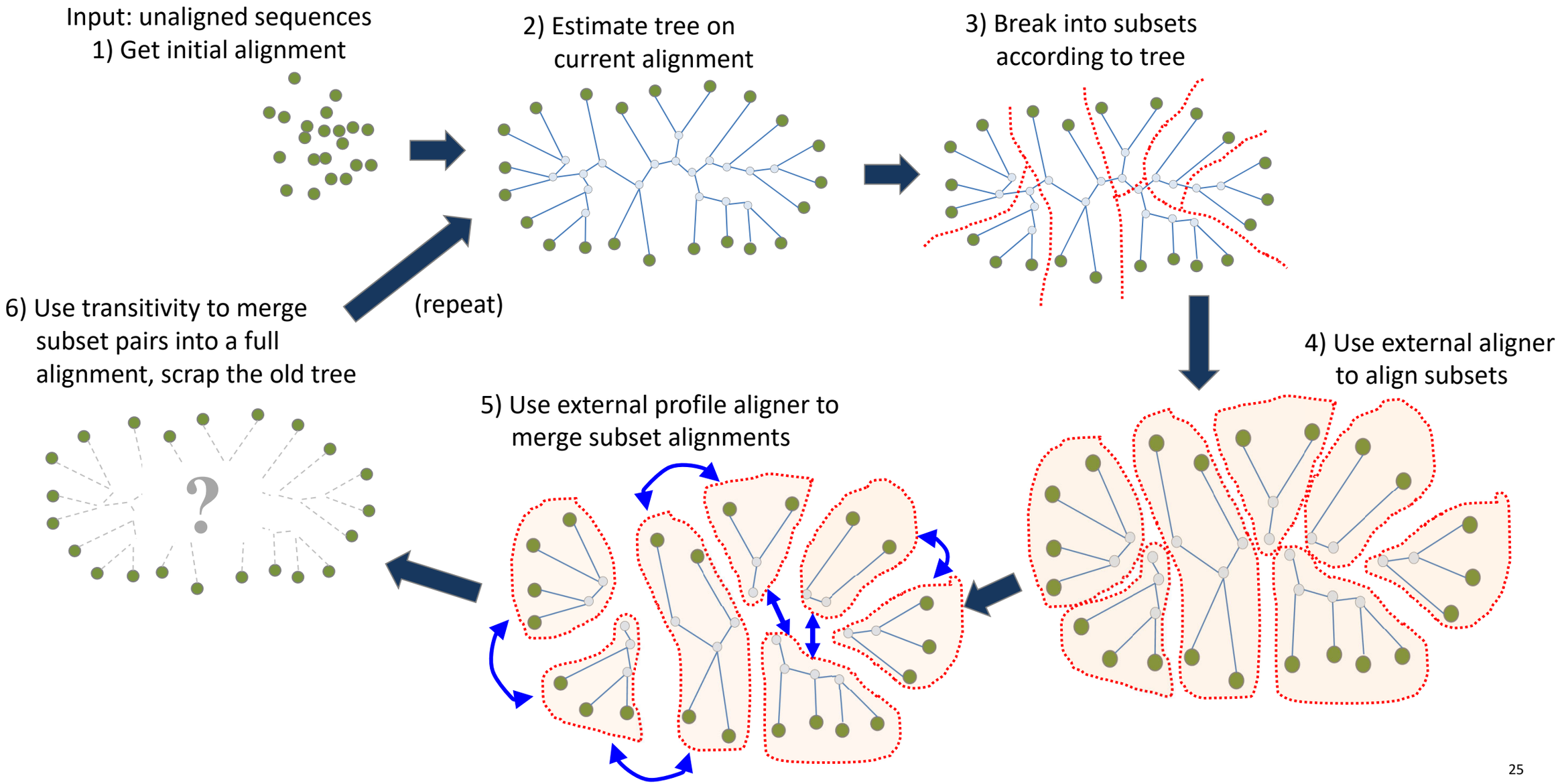

Estimate ML tree on new alignment

**Use tree to compute new alignment**

Repeat until termination condition, and

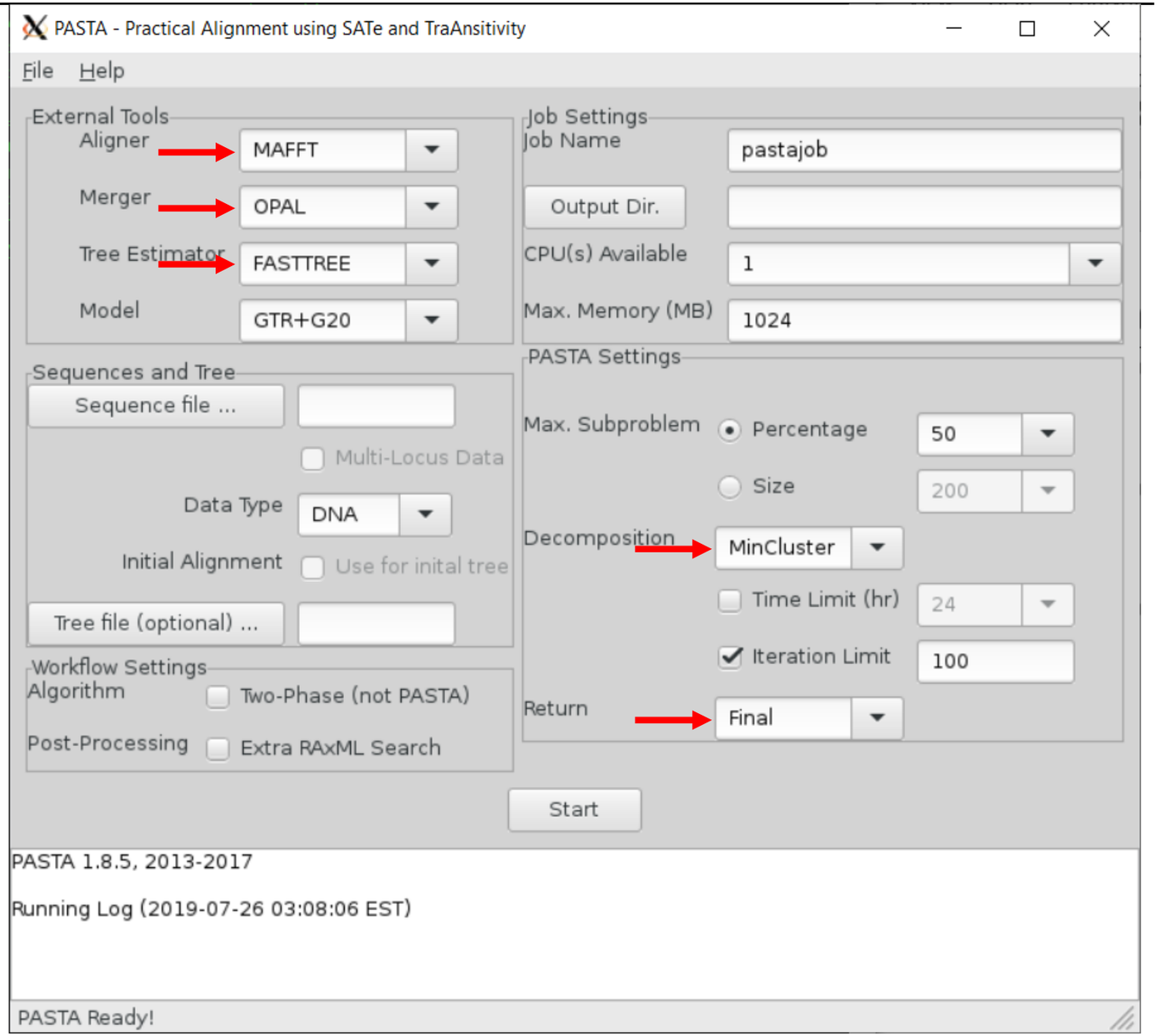
return the alignment/tree pair with the best ML score

# PASTA Algorithm

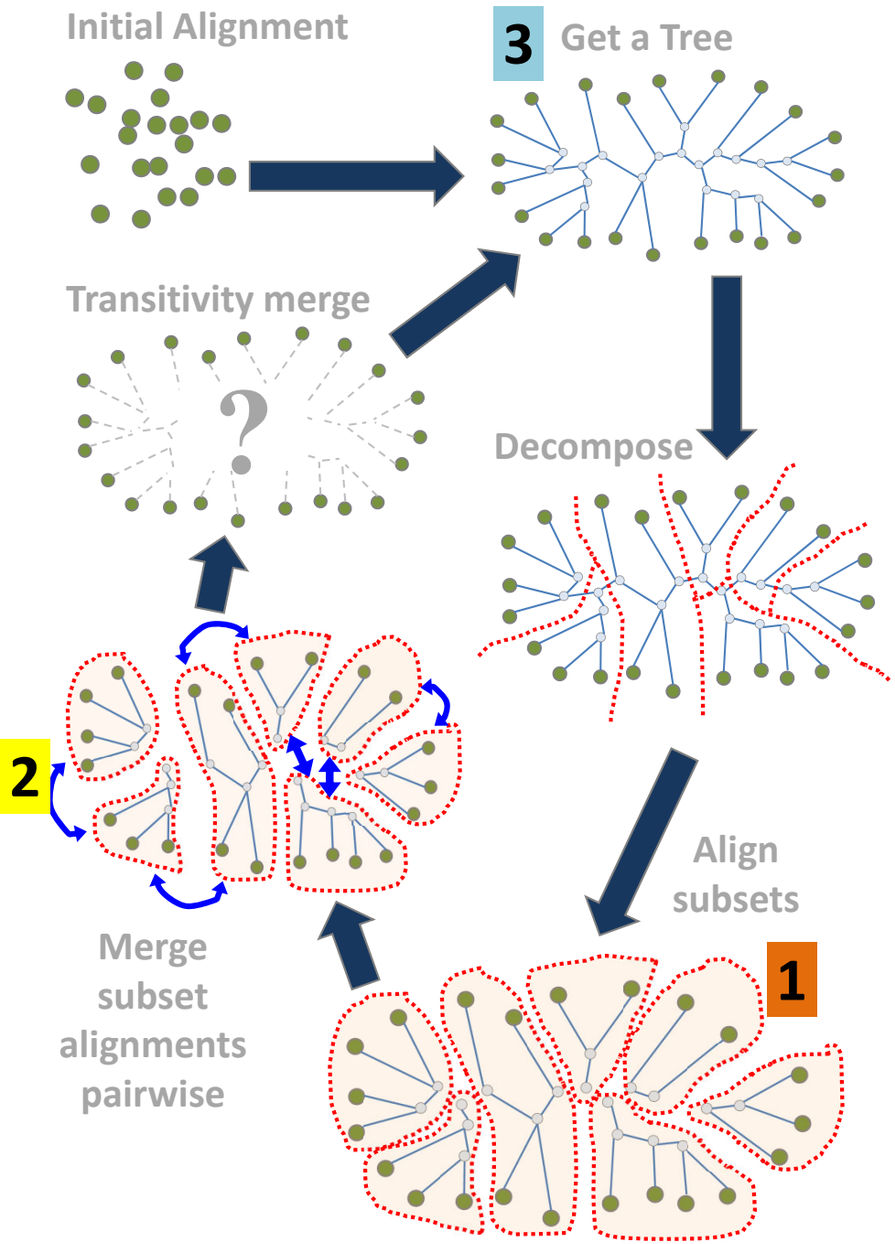


# PASTA GUI (Linux-on-Windows version, but Mac looks the same)

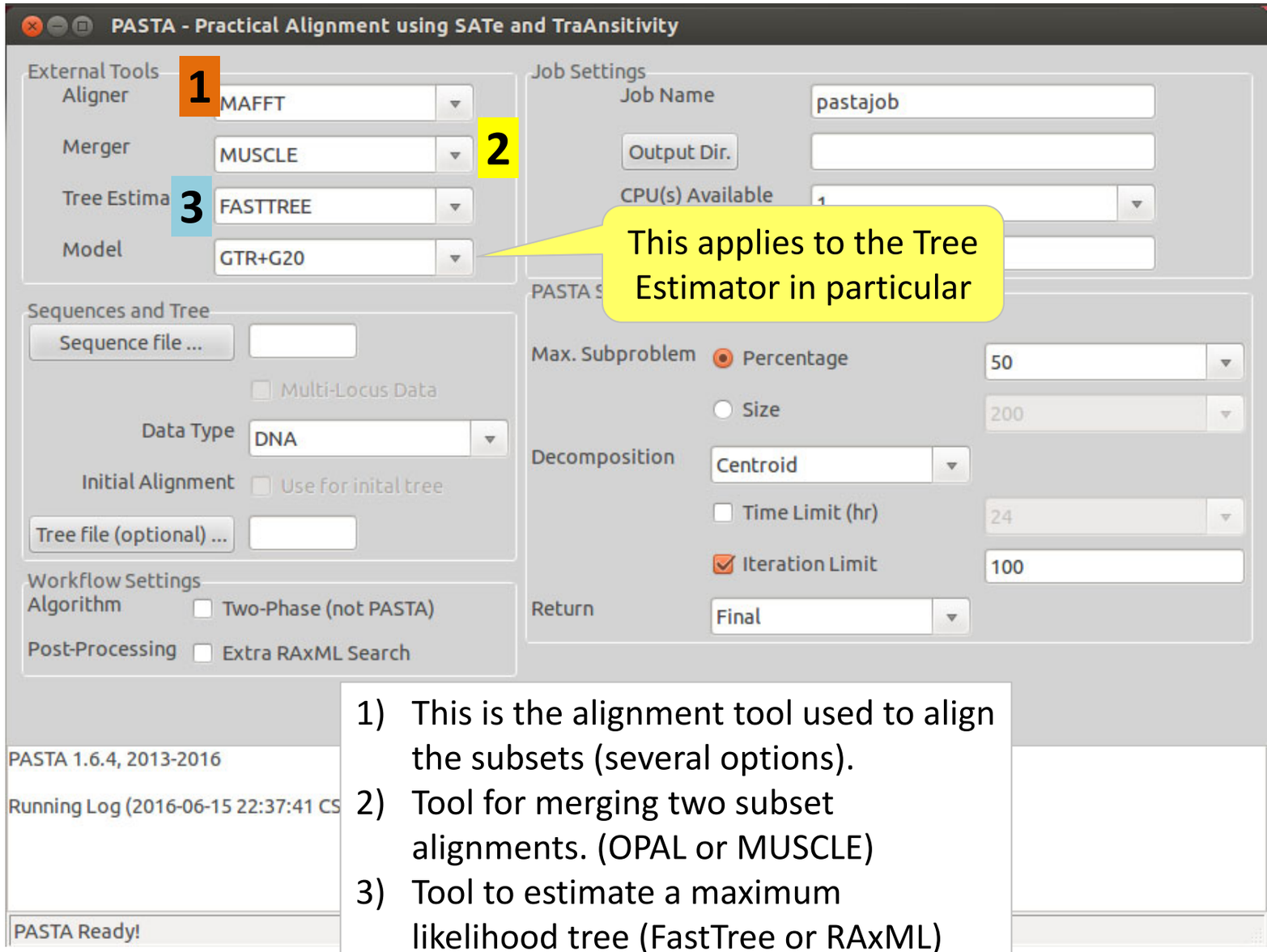
- The GUI is nice because if you import your data by choosing “Sequence File” it will populate many of the options to the optimal defaults for you.
- **I recommend making sure that the things with red arrows are set to what’s shown here, even after importing.**



# PASTA Algorithm

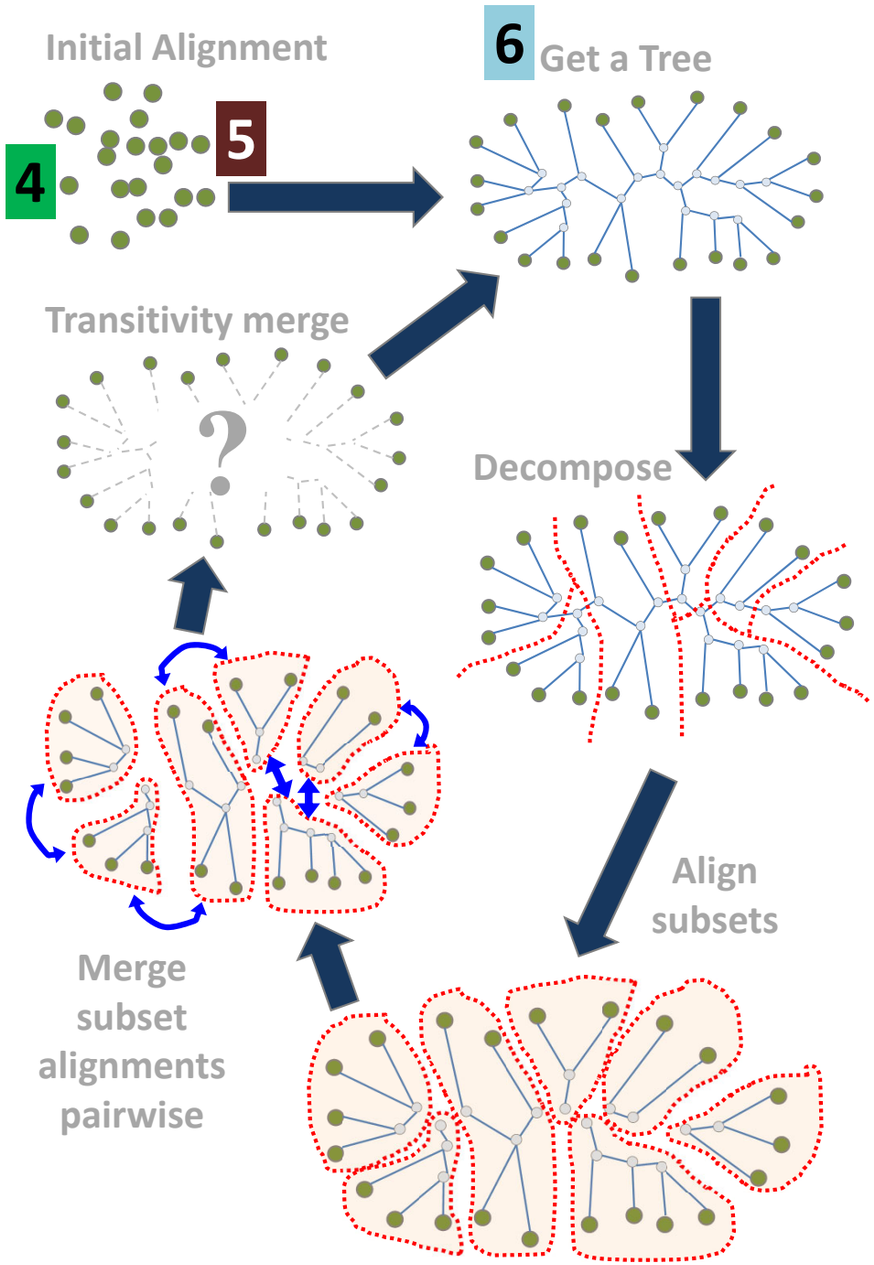


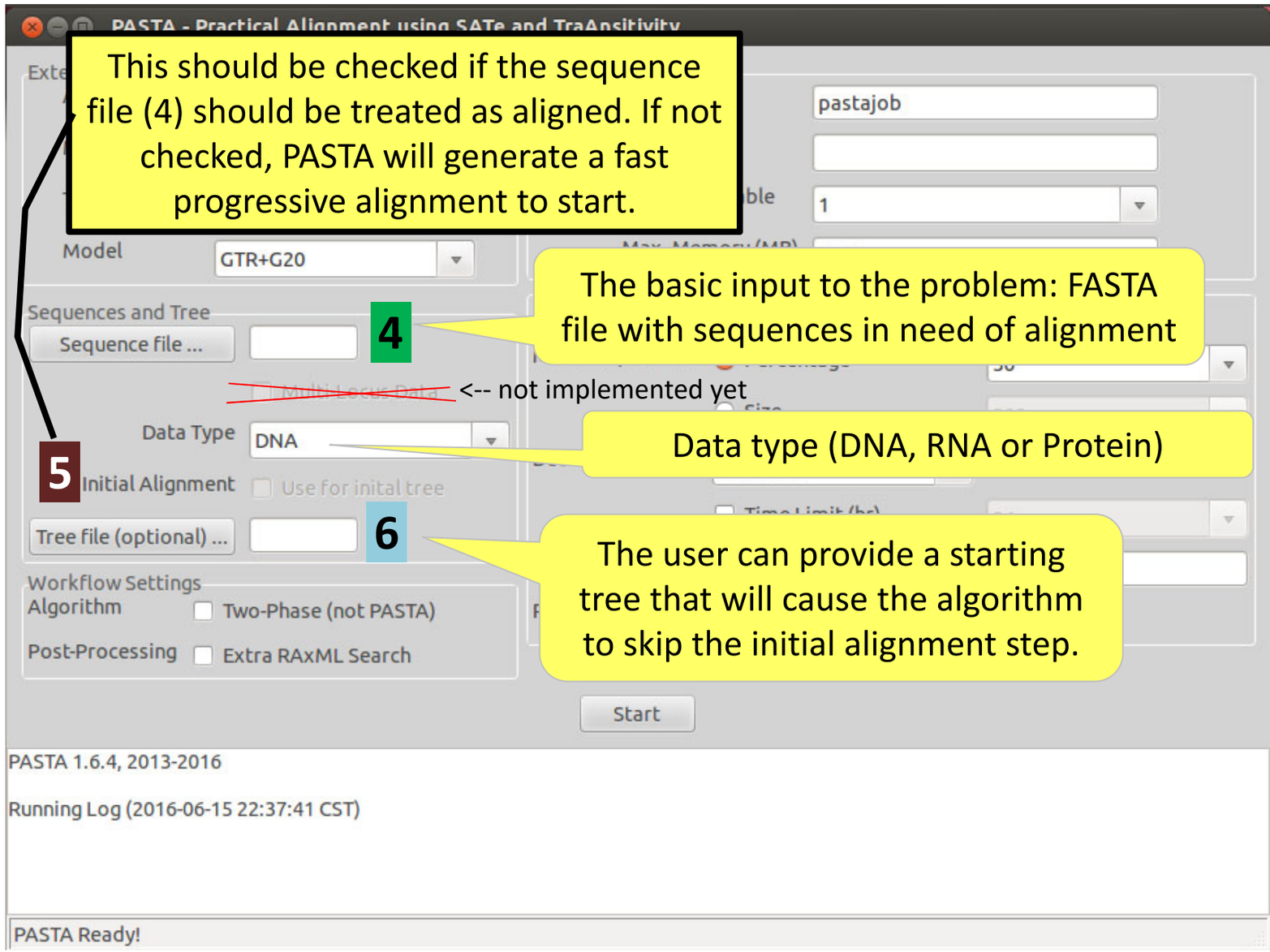

# PASTA GUI



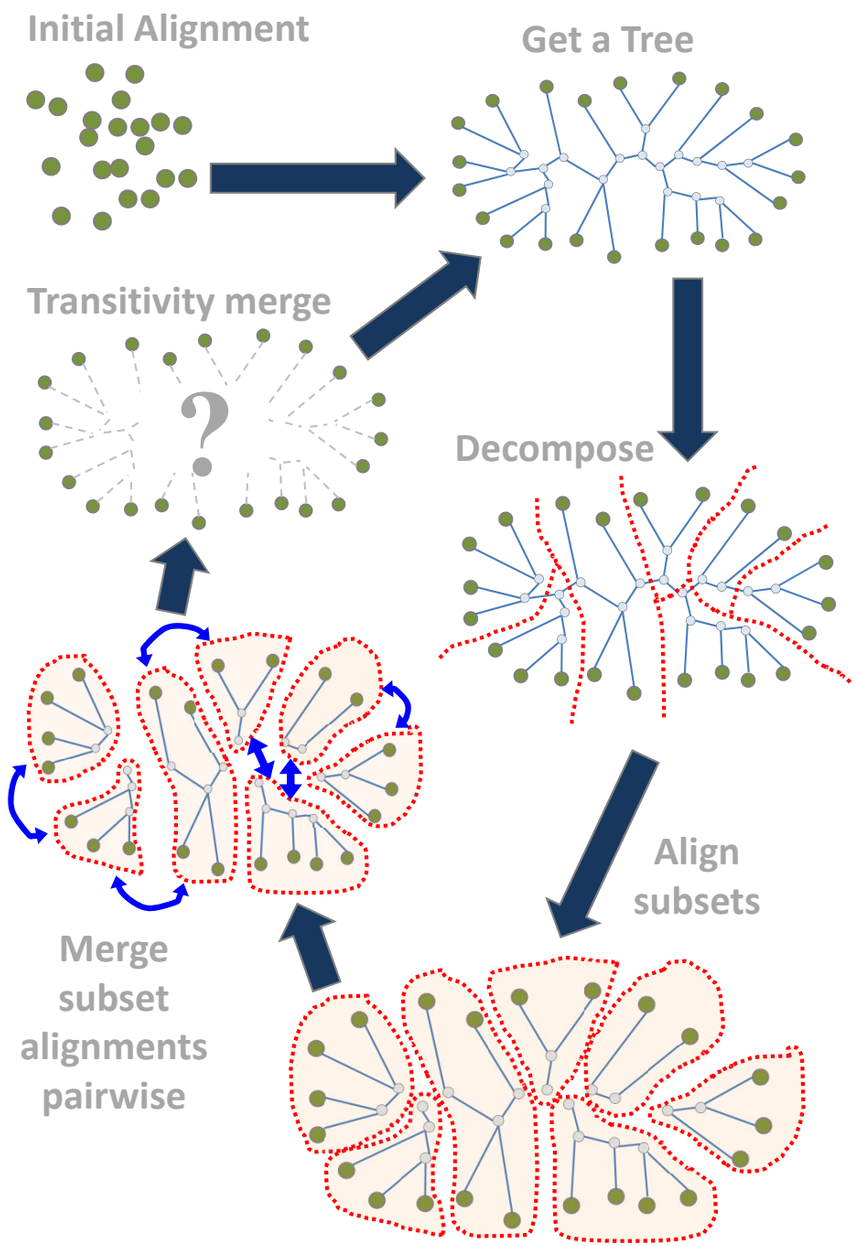

1) This is the alignment tool used to align the subsets (several options).
2) Tool for merging two subset alignments. (OPAL or MUSCLE)
3) Tool to estimate a maximum likelihood tree (FastTree or RAxML)

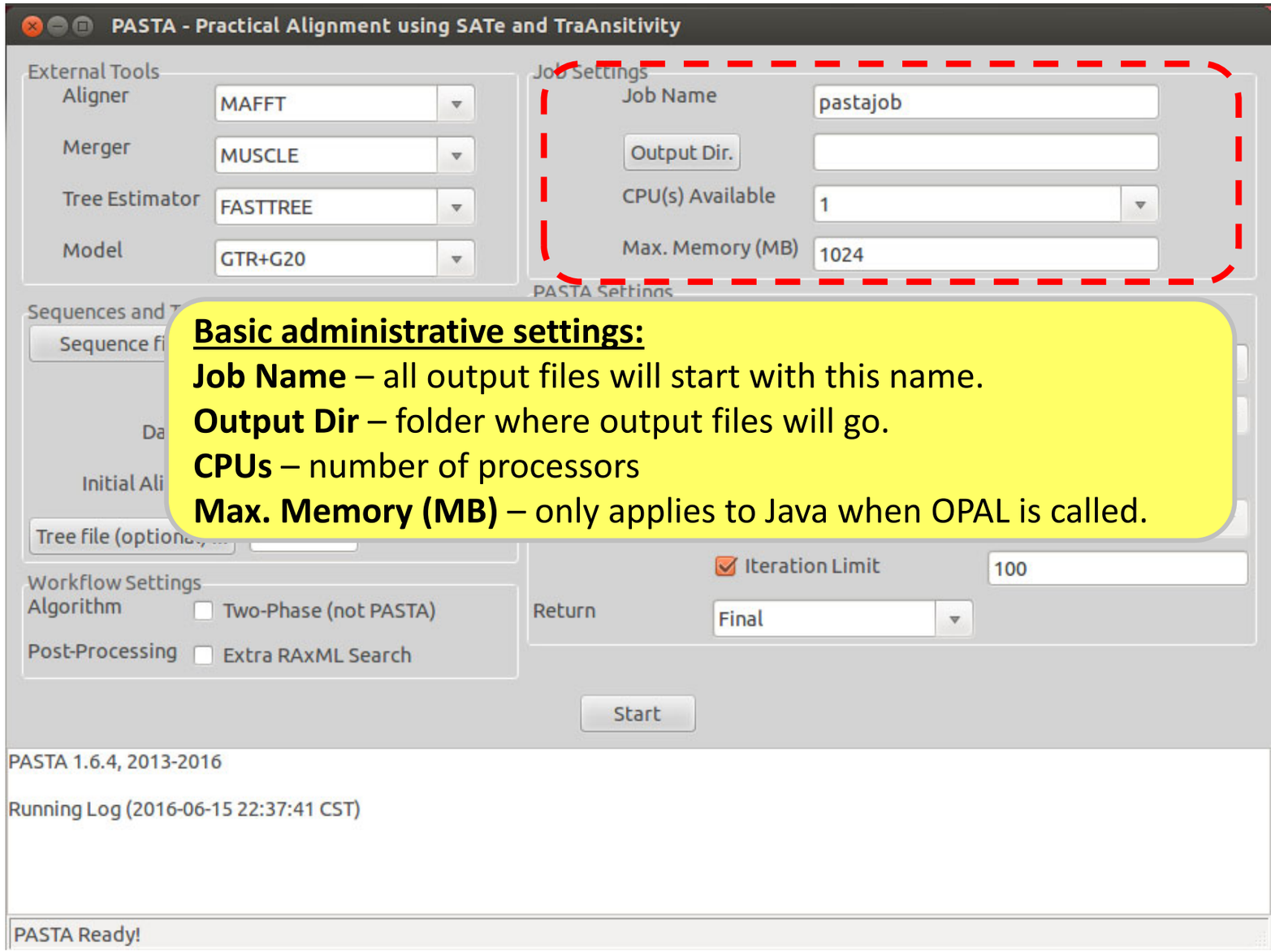
# PASTA Algorithm



# PASTA Algorithm




**Basic administrative settings:**

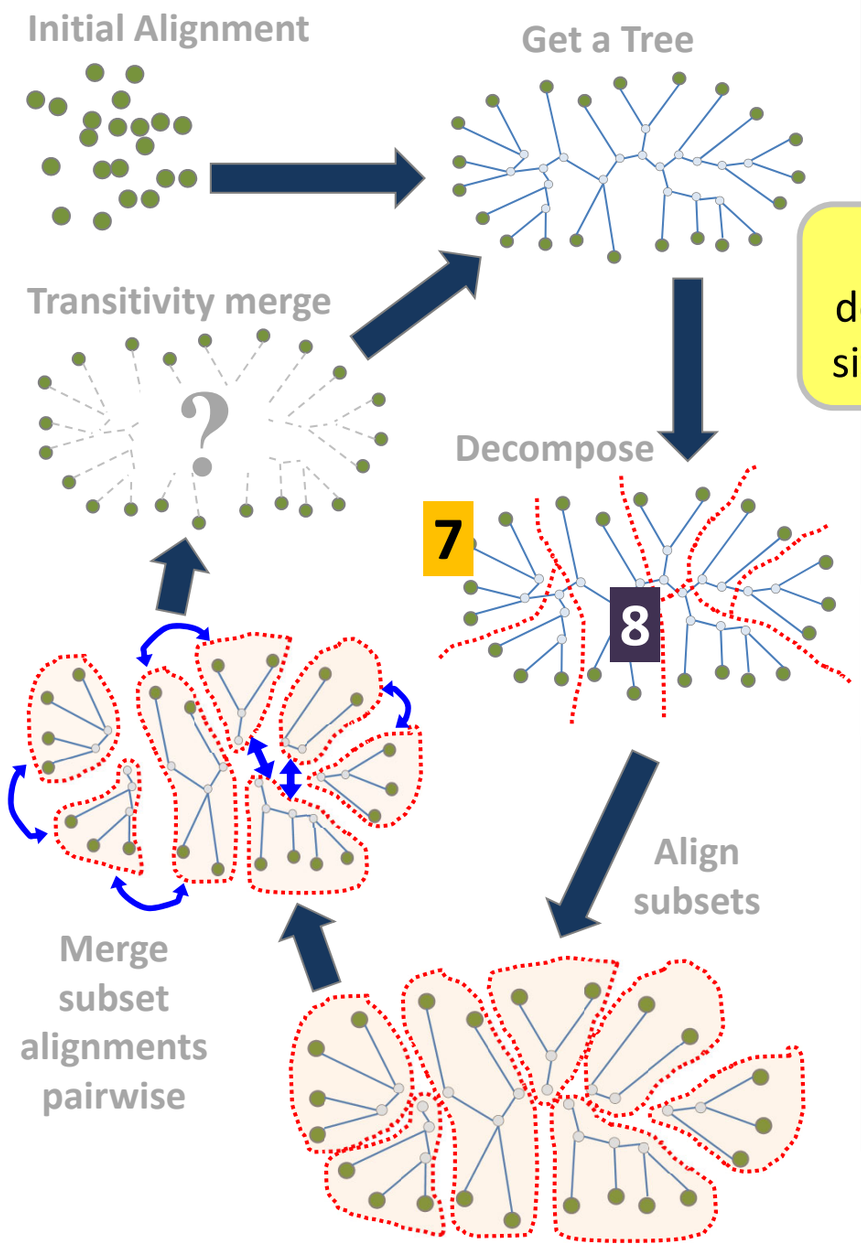
**Job Name** – all output files will start with this name.

**Output Dir** – folder where output files will go.

**CPUs** – number of processors

**Max. Memory (MB)** – only applies to Java when OPAL is called.

# PASTA Algorithm

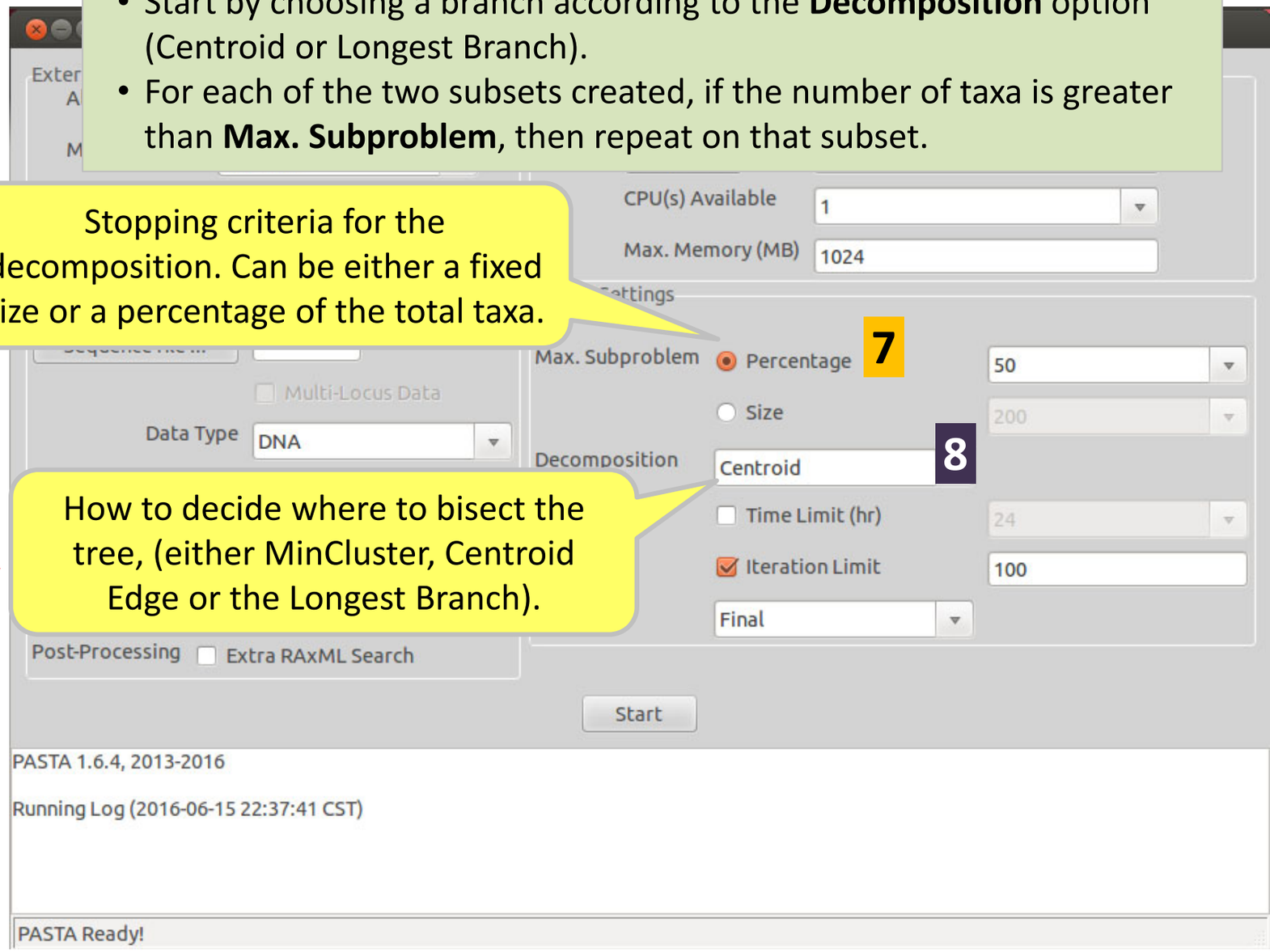



**Decomposition Steps:**

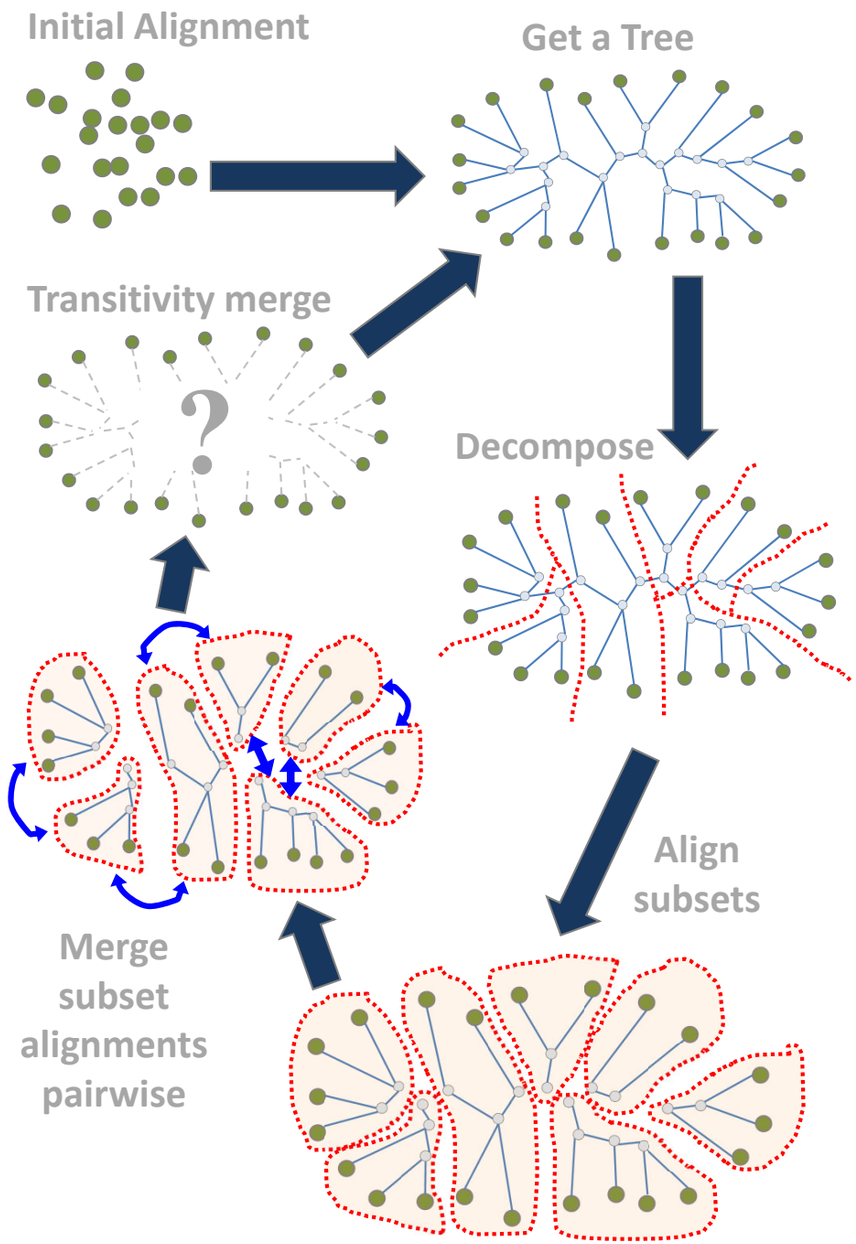
- Start by choosing a branch according to the **Decomposition** option (Centroid or Longest Branch).
- For each of the two subsets created, if the number of taxa is greater than **Max. Subproblem**, then repeat on that subset.

Stopping criteria for the decomposition. Can be either a fixed size or a percentage of the total taxa.

How to decide where to bisect the tree, (either MinCluster, Centroid Edge or the Longest Branch).



# PASTA Algorithm




PASTA - Practical Alignment using SATe and TraAnsitivity

External Tools
Aligner: MAFFT
Merger: MUSCLE
Tree Estimator: FASTTREE
Model: GTR+G20

Job Settings
Job Name: pastajob
Output Dir.
CPU(s) Available: 1
Max. Memory (MB): 1024

Sequences and Tree
Sequence file ...
Multi-Locus Data
Data Type: DNA
Initial Alignment: Use for inital tree
Tree file (optional) ...

PASTA Settings
Max. Subproblem: Percentage 50 / Size 200
Decomposition: Centroid
Time Limit (hr) 24
Iteration Limit 100
Return: Final

Workflow Settings
Algorithm: Two-Phase (not PASTA)
Post-Processing: Extra RAxML Search

Start

PASTA 1.6.4, 2013-2016
Running Log (2016-06-15 22:37:41 CST)
PASTA Ready!

When to Stop Running?

(see below)

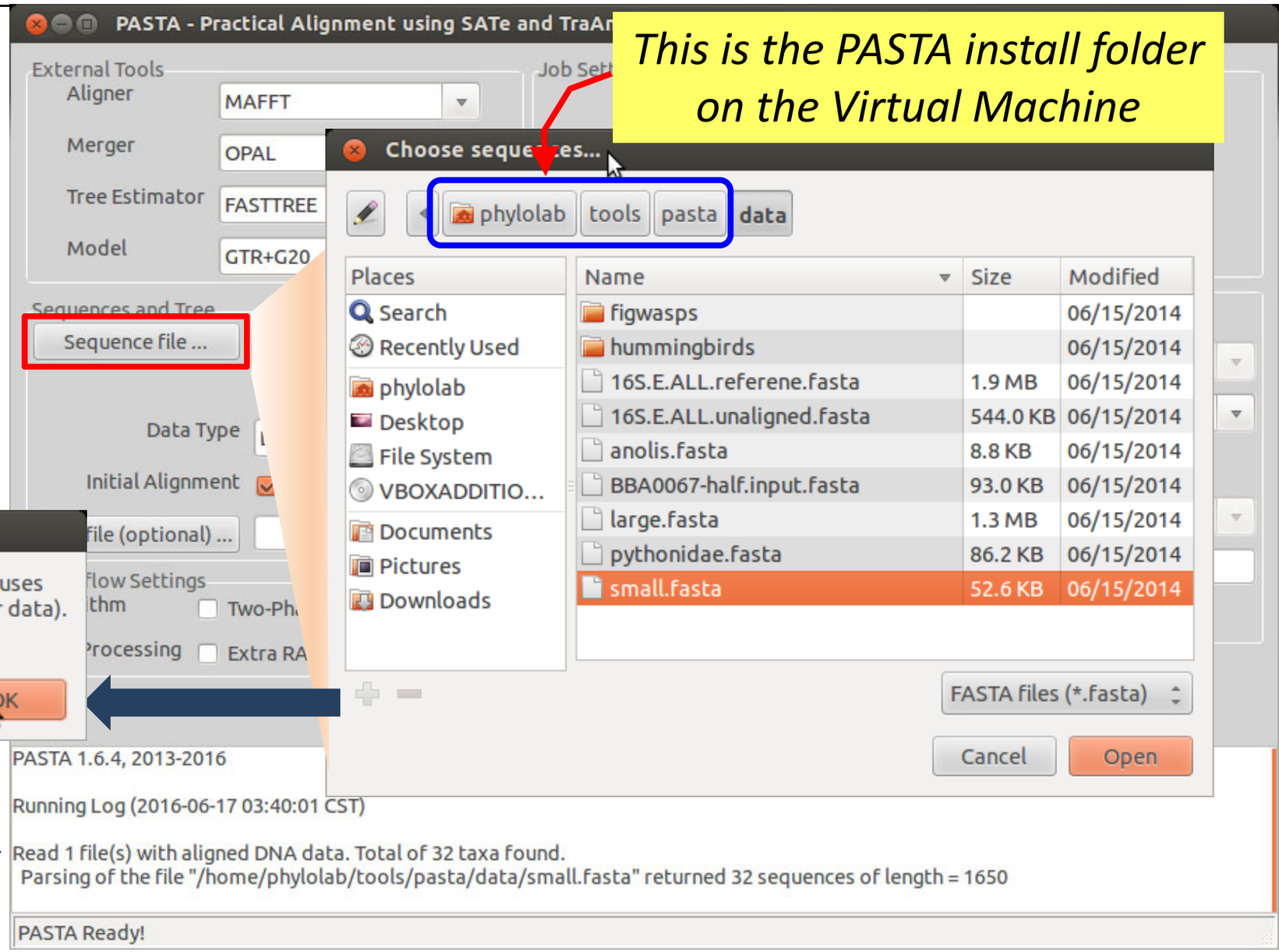
Which iteration to return? (Final or Highest Likelihood)

Should final tree be RAxML?

**Two-Phase** search is simply 1) run an alignment, 2) get a tree from it. This is completely different than PASTA and *if this is checked, PASTA (formally) will not be run*.

# Example 1: small.fasta

**Step 1:** Read in the data. Located at `<pasta-folder>/data/small.fasta`

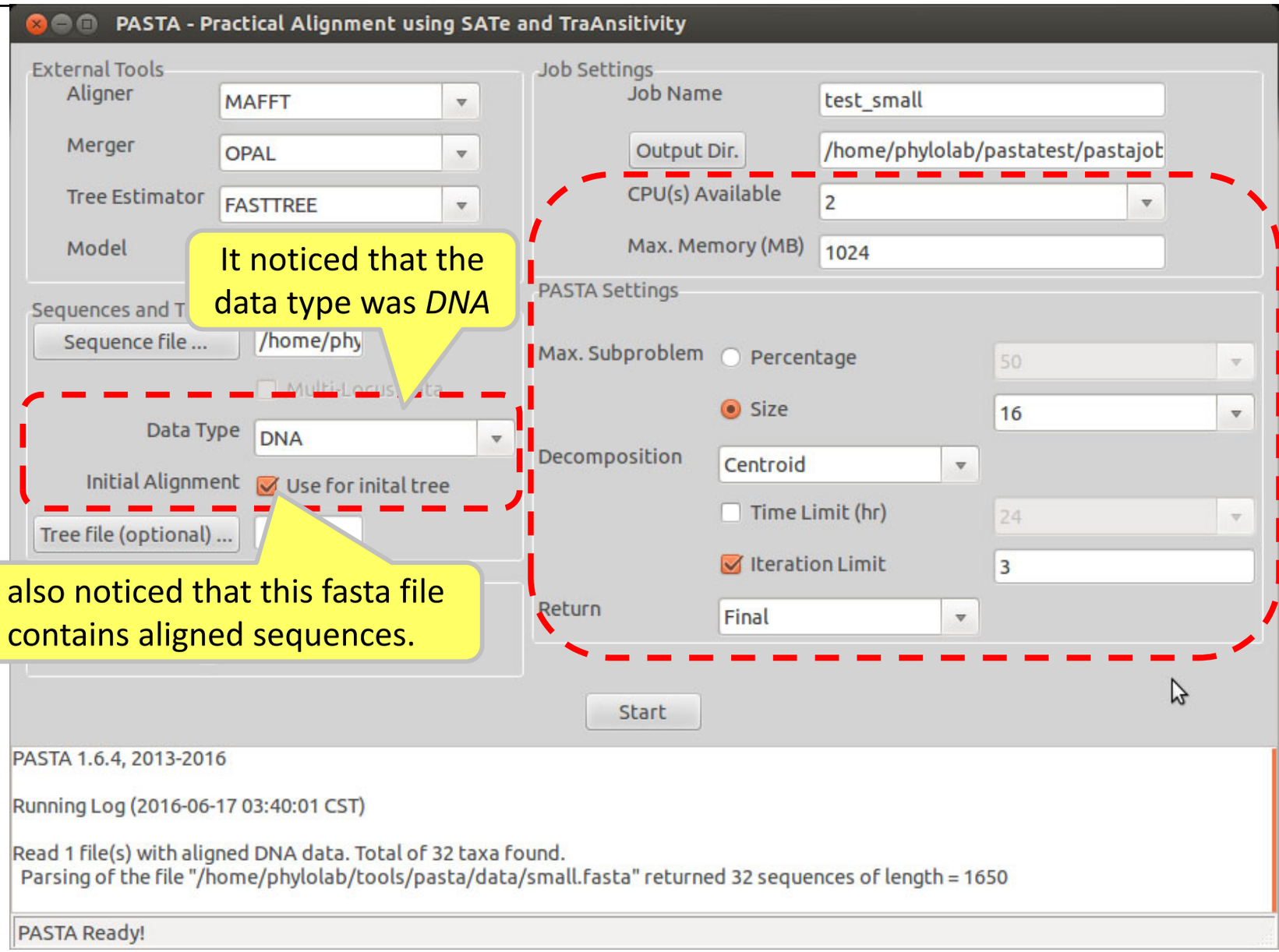



*This is the PASTA install folder on the Virtual Machine*

*Reads in the DATA and sets Type, prints some stats:*

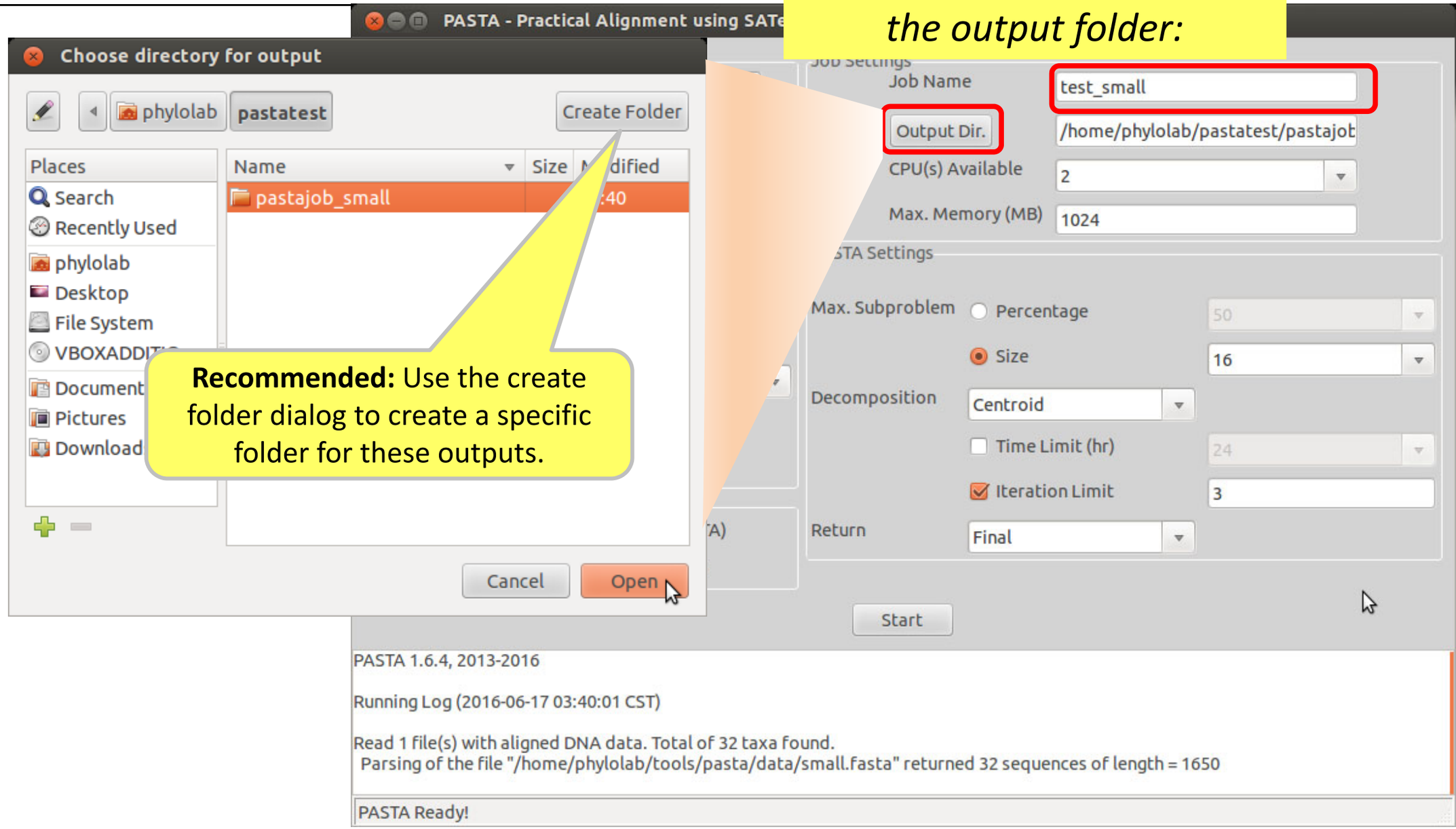
# Example 1: small.fasta

*Importing the data caused the GUI to automatically set several settings based on the size, data type, etc…*



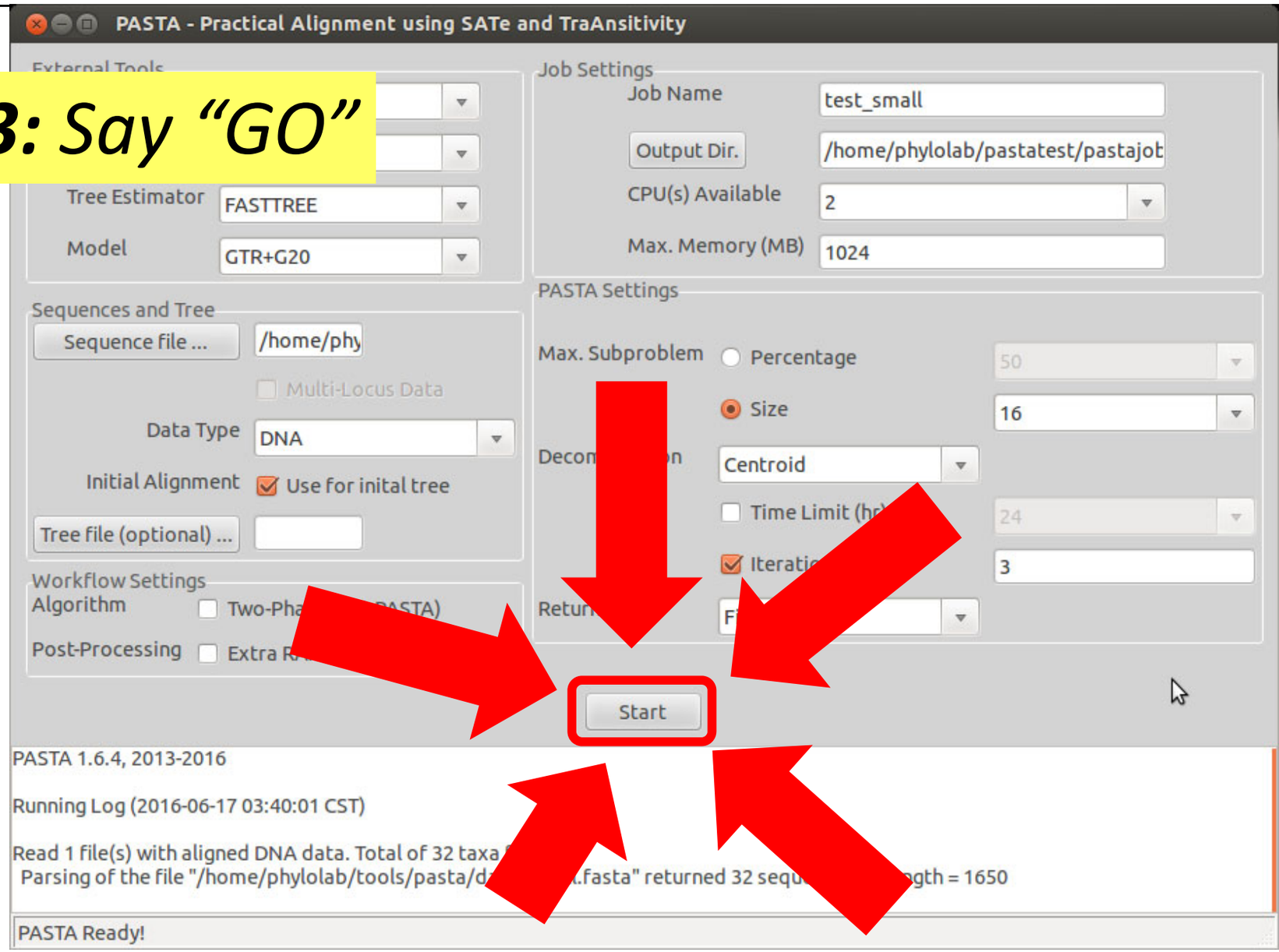
# Example 1: small.fasta

***Step 2:*** *name the job & set the output folder:*



# Example 1: small.fasta

**Step 3:** *Say "GO"*



# Example 1: Examining the Output Folder

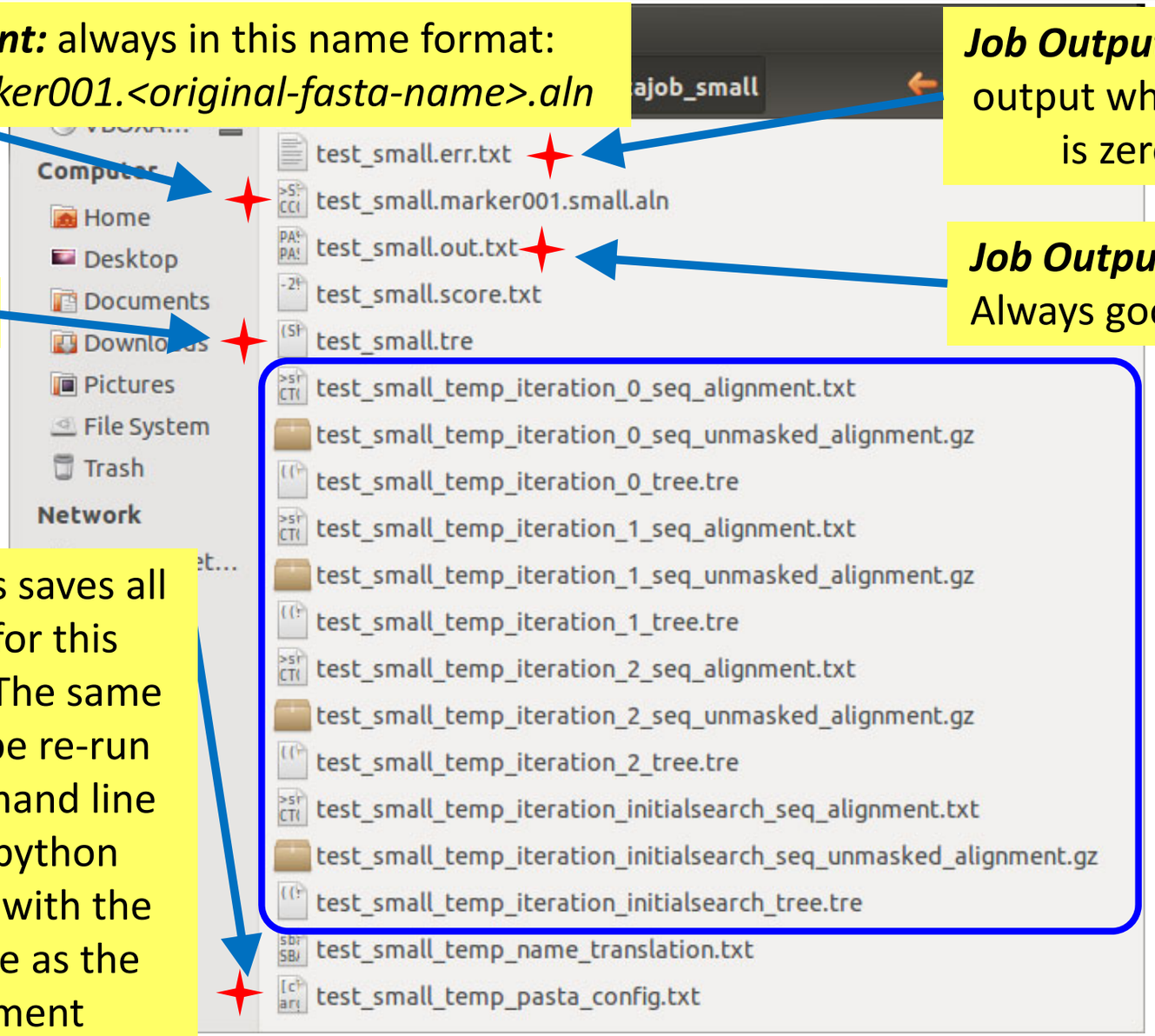
✦ = Important File

***Final Alignment:*** always in this name format: *<jobname>.marker001.<original-fasta-name>.aln*

***Job Output (Errors):*** contains PASTA console output when errors are reported. If this file is zero bytes, that is a good thing.

***Job Output:*** contains PASTA console output. Always good to examine this file after a run.

***Final Tree***




***Config File:*** This saves all the settings for this particular job. The same exact job can be re-run from the command line by running "python run_pasta.py" with the path to this file as the ONLY argument

Intermediate alignments and trees after the initial search and after each iteration. Useful mainly for diagnostics and debugging

# Final Tips & Best Practices

- After running an alignment, it is always a good idea to look at the console outputs generated to verify that PASTA did what it was expected to do. If the error file is non-zero size, read that too.
- The PASTA default settings are appropriate and well-chosen for most applications. Unless you have a good reason to use something else, this is a good starting point.
- PASTA scales with the number of cores available, so giving it as many processors as possible is a good idea.
- There are more settings available than what is in the GUI. Check the config file output for any pasta job to see the full list. Also can type “python run_pasta.py –h” from the pasta folder to see a thorough help menu
- Approximate running time benchmarks (length=1500 base pairs):
  - 100 Sequences: <10 minutes on a laptop
  - 1000 Sequences: About 1-3 hours on a 16-core server
  - 10000 Sequences: About 8-15 hours on a 16-core server
  - (Should scale about linearly after this, but will depend on settings…)