Getting Sequences from GenBank using R-packages

• Open R and select the working directory where you want to output sequence files

Misc>Change Working Directory>select a folder (e.g., R_class_winter_2015)

Alternatively:

```
setwd("/Users/jcsantos/Desktop/R_class_winter_2015/1_getting_sequences_from_GenBank")

#I open terminal and drag the folder to it to get the path. Then, copy and paste.
```

• We need to install and load the following packages:

```
install.packages("ape")
install.packages("seqinr")

library(ape) #this is a general R-package for phylogenetics and comparative methods
library("seqinr") #this is an specialized package for nucleotide sequence management
```

• Let's check that our packages have been loaded correctly

```
sessionInfo()
```

Getting Sequences from GenBank using R-packages

- Let's use '`ape`' to read the sequence from GenBank this with the function: `?read.GenBank`

- This function connects to the GenBank database, and reads nucleotide sequences using accession numbers given as arguments.

- Usage (do not run)

```
read.GenBank(access.nb, seq.names = access.nb, as.character = FALSE)

#access.nb:  a vector of mode character giving the accession numbers.
#seq.names: the names to give to each sequence; by default the accession numbers.
#as.character: a logical whether to return the sequences as an object "DNAbin".
```

- Let's read the casque-headed lizard (*Basiliscus basiliscus*) RAG1 sequence JF806202

```
seq_1_DNAbin <- read.GenBank("JF806202") #save as DNAbin object:
attr(seq_1_DNAbin, "species") #to get the specie name of the sequence
seq_1_DNAbin$JF806202
str(seq_1_DNAbin) # we get the structure of the object

#save as character object:

seq_1_character <- read.GenBank("JF806202", as.character = TRUE)
seq_1_character #this is not a very nice format
```

Read sequences using accession numbers

- Create a vector of GenBank accession numbers that we want

```
lizards_accession_numbers <- c("JF806202", "HM161150", "FJ356743", "JF806205",
                               "JQ073190", "GU457971", "FJ356741", "JF806207",
                               "JF806210", "AY662592", "AY662591", "FJ356748",
                               "JN112660", "AY662594", "JN112661", "HQ876437",
                               "HQ876434", "AY662590", "FJ356740", "JF806214",
                               "JQ073188", "FJ356749", "JQ073189", "JF806216",
                               "AY662598", "JN112653", "JF806204", "FJ356747",
                               "FJ356744", "HQ876440", "JN112651", "JF806215",
                               "JF806209")
#create a vector a GenBank accession numbers
```

- Get those sequences and save them in a single DNAbin object:

```
lizards_sequences <- read.GenBank(lizards_accession_numbers) #read sequences and place
them in a DNAbin object

lizards_sequences #a brief summary of what is in the object, including base composition


str(lizards_sequences) #a list of the DNAbin elements with length of the sequences
                       #notice the one of the attributes is the species names
```

Read sequences and create a fasta file format

- Lets explore more the DNAbin object:

```
attributes(lizards_sequences) #see the list of attributes and contents

names(lizards_sequences) #the accession numbers

attr(lizards_sequences, "species") # we get the species list. Notice this
                                   # attr is slightly different function
```

- However, it is hard remember which accession number corresponds to which species. So we can use the previous information to create first a vector with such information

```
lizards_sequences_GenBank_IDs <- paste(attr(lizards_sequences, "species"), names
(lizards_sequences), sep ="_RAG1_")

## build a character vector with the species, GenBank accession numbers, and gene
## name "_RAG1_" this is its common abbreviation: recombination activating protein 1
## notice the use of the paste function: textA, textB, textC
## results in: textAtextCtextB

lizards_sequences_GenBank_IDs #a more informative vector of names for our sequences
```

Write a fasta file format

• Let's write sequences to a text file in fasta format using write.dna(). However, only accession numbers are included.

```
?write.dna # This function writes in a file a list of DNA sequences in sequential,
interleaved, or FASTA format.

### we are going to write in fasta format

write.dna(lizards_sequences, file ="lizard_fasta_1.fasta", format = "fasta", append =
FALSE, nbcol = 6, colsep = " ", colw = 10)

########### Some relevant arguments for write.dna()

#x: a list or a matrix of DNA sequences.

#file: a file name specified to contain our sequences

#format: Three choices are possible: "interleaved", "sequential", or "fasta", or any
#unambiguous abbreviation of these.

#append: a logical, if TRUE the data are appended to the file without erasing the data
#possibly existing in the file, otherwise the file is overwritten (FALSE the default).

#nbcol: a numeric specifying the number of columns per row (6 by default)

#colsep: a character used to separate the columns (a single space by default).

#colw: a numeric specifying the number of nucleotides per column (10 by default).
###########
```

Write a fasta file format

• Lets explore our recently created file ‘lizard_fasta_1.fasta’. Drag and drop this file in the text editor

File Path ▾ : ~/Desktop/R_class_winter_2015/1_getting_sequences_from_GenBank/lizard_fast

lizard_fasta_1.fasta

```
>JF806202
tccatgccct ccgaactgct gaaaagtccc tcttgccagg ttaccatcca tttgagtgga
aaccaccctt gaaaaatgtc tccagtaata cagaagtagg cattattgat gggctttcag
gcatacaaca tttggttgat gattacccag ttgacacaat tgcaaagaga tttcgatatg
atgctgcttt ggtttctgcc ttgatggata tggaagaaga catcctagaa ggcctgaaaa
gtcaggacat ggatgactat ctcaagggyc ctttcactgt ggtgattaaa gagtcctgtg
atggaatggg agatgttagt gagaaacatg gctgtggccc agctgtccct gaaaaagcag
ttcgattctc tttcacactc atgagcatct ctgtcactca tggcaatgca agcataagga
tttttgaaga aaataagccc aattcagaac tgtgttgtaa acctttgtgc cttatgctgg
ctgatgaatc agaccatgag acactcacag ccatcctgag tcctcttgtg gcagaaagag
aggccatgaa agacagtgta ctgatacttg atatggctgg aatcccgaga atgttcaaat
tcatatttag aggcactgga tatgatgaaa agcttgtccg tgaagtagag ggccttgaag
cttcaggctc tacttacatc tgcacgctgt gtgatgcaac acgcctggag gcctcacaga
acctgatcct tcattccatc acaaggaatc atgtggaaaa cttagaaagg tacgaggtgt
ggagatccaa cccctatcgt gagactgttg atgaactgcg tgacagagtg aaggggggttt
ctgcaaagcc ttttattgag actgtgcctt cgatagatgc cttgcactgt gacattggca
atgcagctga attttacaag atatttcagt ttgagattgg tgaagtctac aaaaaccgcg
atgcatcaaa agaagagaga aagagatggc agtcagct
>HM161150
aataaaggaa aagtggcagc ttctctggac aaagtcagtg aggaaaagac tgagactgtg
gctgtaaagt cacacccacc ctttgaaaca gacatccagt tgaacaaatg tattcagaaa
atagataagg gtgcctttca tatgagccaa acagaggctg aaacacacca ggtaaacctg
```

• This file has our sequences, but we only have the accession numbers

Rewrite a fasta file format with more information

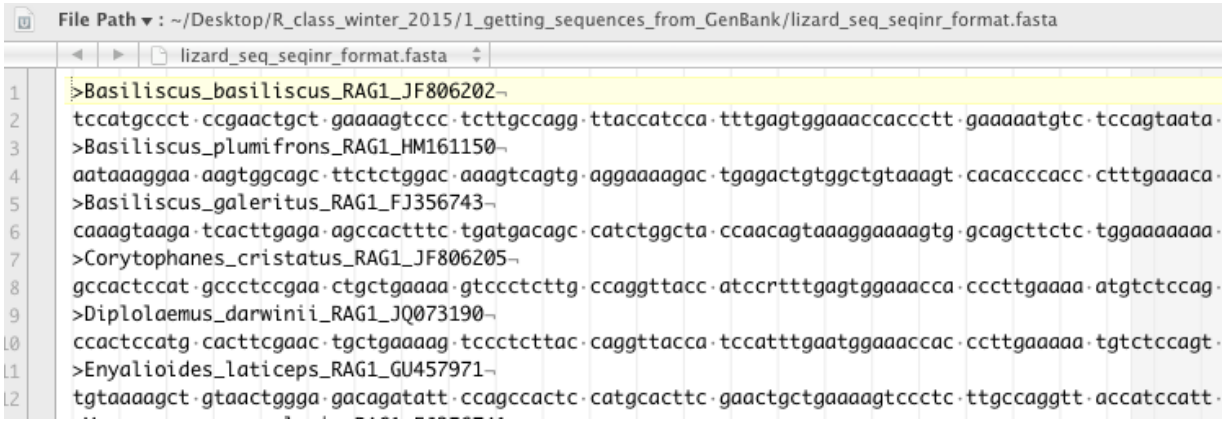
- Read our fasta file using the seqinr package

```
lizard_seq_seqinr_format <- read.fasta(file = "lizard_fasta_1.fasta", seqtype = "DNA",
as.string = TRUE, forceDNAtolower = FALSE)

lizard_seq_seqinr_format #this shows different form to display the same sequence
                         #information
```

- Rewrite our fasta file using the name vector that we created previously

```
write.fasta(sequences = lizard_seq_seqinr_format, names = lizards_sequences_GenBank_IDs,
nbchar = 10, file.out = "lizard_seq_seqinr_format.fasta")

#Suggestion: Do not rearrange, delete or add sequenced to the fasta file, as the
function will assign the names in the order provided in the file and the name vector
```

- Let's check our new fasta file '`lizard_seq_seqinr_format.fasta`'



Get sequences without using accession numbers

• We can use a package that use an API (application programming interface) to interact with the NCBI website.

More info in: http://en.wikipedia.org/wiki/Application_programming_interface

```
install.packages ("rentrez")
library (rentrez)
```

• Let's get some lizard sequences

```
lizard <- "Basiliscus basiliscus[Organism]" #We want a character vector

#nucleotide database (nuccore) and retmax determines the max number
lizard_search <- entrez_search(db="nuccore", term=lizard, retmax=40)
lizard_search
lizard_search$ids #gives you the NCBI ids


#gets your sequences as a character vector
lizard_seqs <- entrez_fetch(db="nuccore", id=lizard_search$ids, rettype="fasta")
lizard_seqs
```

Get sequences without using accession numbers

- Lets get our *Basiliscus basiliscus* RAG 1 sequence

```
Bbasiliscus_RAG1 <- "Basiliscus basiliscus[Organism] AND RAG1[Gene]"

Bbasiliscus_RAG1_search <- entrez_search(db="nuccore", term=Bbasiliscus_RAG1, retmax=10)
#nucleotide database (nuccore) and retmax determines no more than 10 access numbers to
return

Bbasiliscus_RAG1_search$ids #gives you the NCBI ids

Bbasiliscus_RAG1_seqs <- entrez_fetch(db="nuccore", id=Bbasiliscus_RAG1_search$ids,
rettype="fasta")

Bbasiliscus_RAG1_seqs #notice \n (new line) delimiter. Other common delimiters are \r
                      #(carriage return) and \t (tab).

write(Bbasiliscus_RAG1_seqs, "Bbasiliscus_RAG1.fasta", sep="\n") #gets sequence to a
file
```

- We can read our fasta file using seqinr package

```
Bbasiliscus_RAG1_seqinr_format <- read.fasta(file = "Bbasiliscus_RAG1.fasta", seqtype =
"DNA", as.string = TRUE, forceDNAtolower = FALSE)

Bbasiliscus_RAG1_seqinr_format # you can also check the .fasta file in the working
folder
```

Example: Accessing Cytochrome B Sequences

• We can use the `'rentrez'` package to get lots of sequences using taxonomic classifications for specific markers

```
Liolaemus_CYTB <- "Liolaemus[Organism] AND CYTB[Gene]"

#This is a well-studied gene from this genus of South American lizards

Liolaemus_CYTB_search <- entrez_search(db="nuccore", term=Liolaemus_CYTB, retmax=100)

Liolaemus_CYTB_search #There are 2539 sequences that match this query
```

• Let's adjust the search and fetch all sequences of of sequences using taxonomic classifications for specific markers

```
Liolaemus_CYTB_search_2 <- entrez_search(db="nuccore", term=Liolaemus_CYTB, retmax=2539)

Liolaemus_CYTB_search_2$ids #gives you the NCBI ids

Liolaemus_CYTB_seqs <- entrez_fetch(db="nuccore", id=Liolaemus_CYTB_search_2$ids ,
rettype="fasta")

#we get an error "client error: (414) Request-URI Too Long". We are asking too many
sequences
```

Example: Accessing Cytochrome B Sequences

• Lets adjust the search and fetch by smaller chunks so we can get the first 1500 sequences

```
Liolaemus_CYTB_seqs_part_1 <- entrez_fetch(db="nuccore", id=Liolaemus_CYTB_search_2$ids
[1:500] , rettype="fasta")

Liolaemus_CYTB_seqs_part_2 <- entrez_fetch(db="nuccore", id=Liolaemus_CYTB_search_2$ids
[501:1000] , rettype="fasta")

Liolaemus_CYTB_seqs_part_3 <- entrez_fetch(db="nuccore", id=Liolaemus_CYTB_search_2$ids
[1001:1500] , rettype="fasta")
```

• Lets write as single file by appending all 3 chucks of sequences

```
write(Liolaemus_CYTB_seqs_part_1, "Liolaemus_CYTB_seqs.fasta", sep="\n")

write(Liolaemus_CYTB_seqs_part_2, "Liolaemus_CYTB_seqs.fasta", sep="\n", append = TRUE)
#it gets the sequences to the same file by changing the logical argument of append from
#the default FALSE to TRUE (i.e., can abbreviate TRUE with T or other unambiguous
#abbreviation)

write(Liolaemus_CYTB_seqs_part_3, "Liolaemus_CYTB_seqs.fasta", sep="\n", append = TRUE)
#you will get a 1.3 Mb file with all 1500 sequences
```

Example: Accessing Cytochrome B Sequences

- We can read our fasta file using the seqinr package and rename the sequences

```
Liolaemus_CYTB_seqs_seqinr_format <- read.fasta(file = "Liolaemus_CYTB_seqs.fasta",
seqtype = "DNA", as.string = TRUE, forceDNAtolower = FALSE)

Liolaemus_CYTB_seqs_seqinr_format

Liolaemnus_CYTB_names <- attr(Liolaemus_CYTB_seqs_seqinr_format, "name")

Liolaemnus_CYTB_names <- gsub("\\..*","", Liolaemnus_CYTB_names)

#eliminate characters after "." using ?gsub (Pattern Matching and Replacement)

Liolaemnus_CYTB_names <- gsub("^.*\\|", "", Liolaemnus_CYTB_names)

#eliminate characters before "|" using ?gsub (Pattern Matching and Replacement)

Liolaemnus_CYTB_names
```

Example: Accessing Cytochrome B Sequences

• We can read our fasta file using ape package to get accession numbers and species names

```
Liolaemus_CYTB_seqs_ape_format <- read.GenBank(Liolaemnus_CYTB_names)

attr(Liolaemus_CYTB_seqs_ape_format, "species")
#to get the species names of the sequence

names(Liolaemus_CYTB_seqs_ape_format)

Liolaemus_CYTB_seqs_GenBank_IDs <- paste(attr(Liolaemus_CYTB_seqs_ape_format,
"species"), names(Liolaemus_CYTB_seqs_ape_format), sep="_CYTB_")
## build a vector object with the species, GenBank accession numbers, and type of gene

Liolaemus_CYTB_seqs_GenBank_IDs #vector of names to add to sequences

# Read our fasta file 'Liolaemus_CYTB_seqs.fasta' using seqinr package

Liolaemus_CYTB_seqs_seqinr_format <- read.fasta(file = "Liolaemus_CYTB_seqs.fasta",
seqtype = "DNA", as.string = TRUE, forceDNAtolower = FALSE)

# Rewrite our fasta file using the name vector that we created previously

write.fasta(sequences = Liolaemus_CYTB_seqs_seqinr_format, names =
Liolaemus_CYTB_seqs_GenBank_IDs, nbchar = 10, file.out =
"Liolaemus_CYTB_seqs_seqinr_format.fasta")
```

Alignment and Simultaneous Tree Estimation

- We are going to use SATe-2 (SATé - Simultaneous Alignment and Tree Estimation)

URL: http://phylo.bio.ku.edu/software/sate/sate.html

Syst. Biol. 61(1):90–106, 2011
© The Author(s) 2011. Published by Oxford University Press, on behalf of the Society of Systematic Biologists. All rights reserved.
For Permissions, please email: journals.permissions@oup.com
DOI:10.1093/sysbio/syr095
Advance Access publication on December 1, 2011

SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees

KEVIN LIU[1], TANDY J. WARNOW[1], MARK T. HOLDER[2], SERITA M. NELESEN[3], JIAYE YU[2], ALEXANDROS P. STAMATAKIS[4], AND C. RANDAL LINDER[5,*]

[1]*Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA;* [2]*Department of Ecology and Evolutionary Biology, University of Kansas at Lawrence, Lawrence, KS 66045, USA;* [3]*Department of Computer Science, Calvin College, Grand Rapids, MI 49546, USA; and* [4]*Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, D-69118 Heidelberg, Germany;* [5]*Section of Integrative Biology, School of Biological Sciences, University of Texas at Austin, Austin, TX, USA;*
**Correspondence to be sent to: The University of Texas at Austin, Austin, TX 78712, USA; E-mail: rlinder@mail.utexas.edu.*

Received 20 August 2010; reviews returned 9 November 2010; accepted 3 June 2011
Associate Editor: Laura Kubatko

Alignment and Simultaneous Tree Estimation

• From the Developers' webpage (University of Kansas: Jiaye Yu, Mark Holder, Jeet Sukumaran, Siavash Mirarab, and Jamie Oaks):

SATé is a software package for inferring a sequence alignment and phylogenetic tree. The iterative algorithm involves repeated alignment and tree searching operations. The original data set is divided into smaller subproblems by a tree-based decomposition. These sub-problems are aligned and further merged for phylogenetic tree inference.

Currently, the following tools are supported, and are bundled with the SATé distribution:

ClustalW 2.0.12 (sequence alignment program)
MAFFT 6.717 (sequence alignment program)
MUSCLE 3.7 (sequence alignment program)
OPAL 1.0.3 (sequence alignment program)

PRANK 100311 (phylogeny-aware alignment program)

RAxML 7.2.6 (phylogeny estimator program)
FastTree 2.1.4 (phylogeny estimator program)

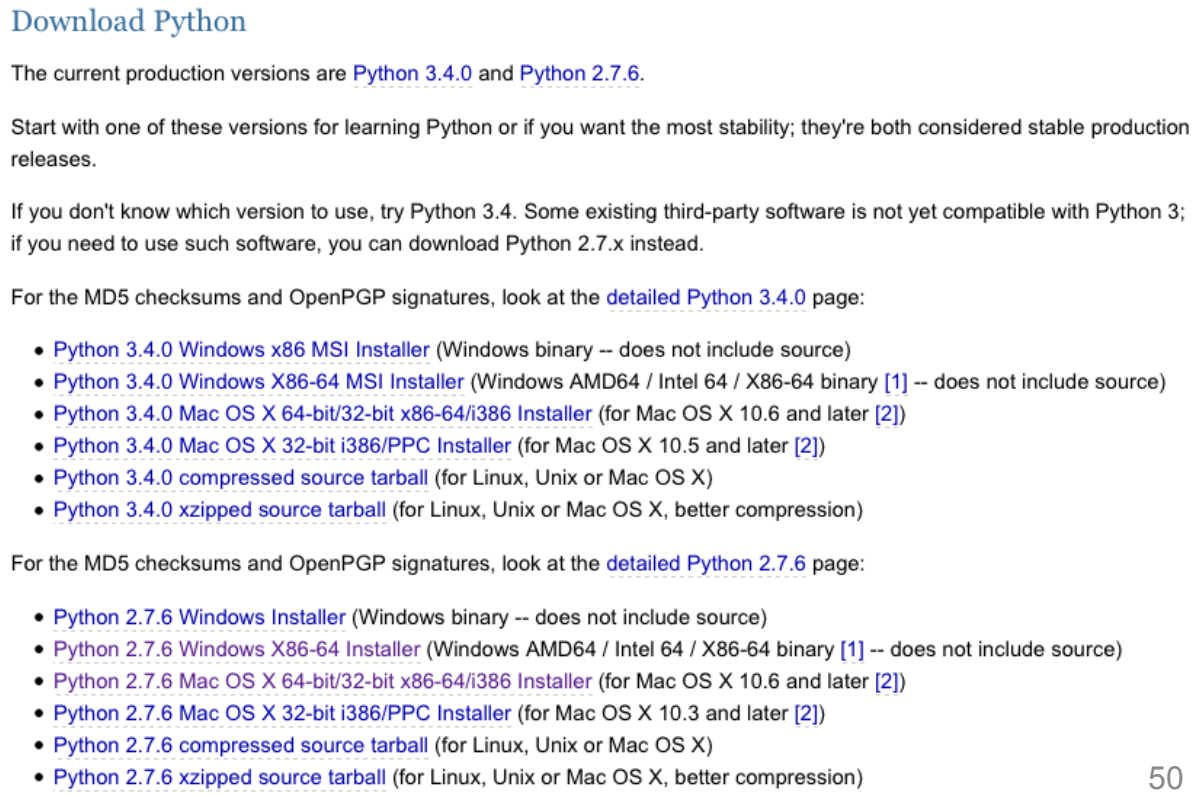
SATe-2 needs Python 2.7 (Upgrade Python Instructions)

- MAC OS: Open terminal (go the HD>Applications>Utilities>Terminal)
- MAC OS: Check your version of Python

```
python --version
```

- MAC OS: if necessary upgrade python to 2.7 as required by SATe-II

http://legacy.python.org/download/



Download Python

The current production versions are Python 3.4.0 and Python 2.7.6.

Start with one of these versions for learning Python or if you want the most stability; they're both considered stable production releases.

If you don't know which version to use, try Python 3.4. Some existing third-party software is not yet compatible with Python 3; if you need to use such software, you can download Python 2.7.x instead.

For the MD5 checksums and OpenPGP signatures, look at the detailed Python 3.4.0 page:

- Python 3.4.0 Windows x86 MSI Installer (Windows binary -- does not include source)
- Python 3.4.0 Windows X86-64 MSI Installer (Windows AMD64 / Intel 64 / X86-64 binary [1] -- does not include source)
- Python 3.4.0 Mac OS X 64-bit/32-bit x86-64/i386 Installer (for Mac OS X 10.6 and later [2])
- Python 3.4.0 Mac OS X 32-bit i386/PPC Installer (for Mac OS X 10.5 and later [2])
- Python 3.4.0 compressed source tarball (for Linux, Unix or Mac OS X)
- Python 3.4.0 xzipped source tarball (for Linux, Unix or Mac OS X, better compression)

For the MD5 checksums and OpenPGP signatures, look at the detailed Python 2.7.6 page:

- Python 2.7.6 Windows Installer (Windows binary -- does not include source)
- Python 2.7.6 Windows X86-64 Installer (Windows AMD64 / Intel 64 / X86-64 binary [1] -- does not include source)
- Python 2.7.6 Mac OS X 64-bit/32-bit x86-64/i386 Installer (for Mac OS X 10.6 and later [2])
- Python 2.7.6 Mac OS X 32-bit i386/PPC Installer (for Mac OS X 10.3 and later [2])
- Python 2.7.6 compressed source tarball (for Linux, Unix or Mac OS X)
- Python 2.7.6 xzipped source tarball (for Linux, Unix or Mac OS X, better compression)

Install SATe-2

• Download SATe-II precompiled from UT-Austin website:

http://phylo.bio.ku.edu/software/sate/downloads2/mac/satemac-v2.2.7-2013Feb15.dmg

Archived bundles of SATé:

File name	Build/Platform	Date	Size (MB)
satesrc-v2.2.7-2013Feb15.tar.gz	src	2013-Feb-15	25.5
satemac-v2.2.7-2013Feb15.dmg	mac	2013-Feb-15	25.9
satewin-v2.2.7-2013Feb15.zip	win	2013-Feb-15	26.6

***For those more adventurous you can download the command based 'SATe-II' from:

http://phylo.bio.ku.edu/software/sate/downloads2/

download: satesrc-v2.2.7-2013Feb15.tar.gz

Follow the instructions in the main webpage

Preparing FASTA filed for SATe-2

• Download the FASTA files from the course website

Liolaemus_CYTB.fasta
Lizard_RAG1.fasta

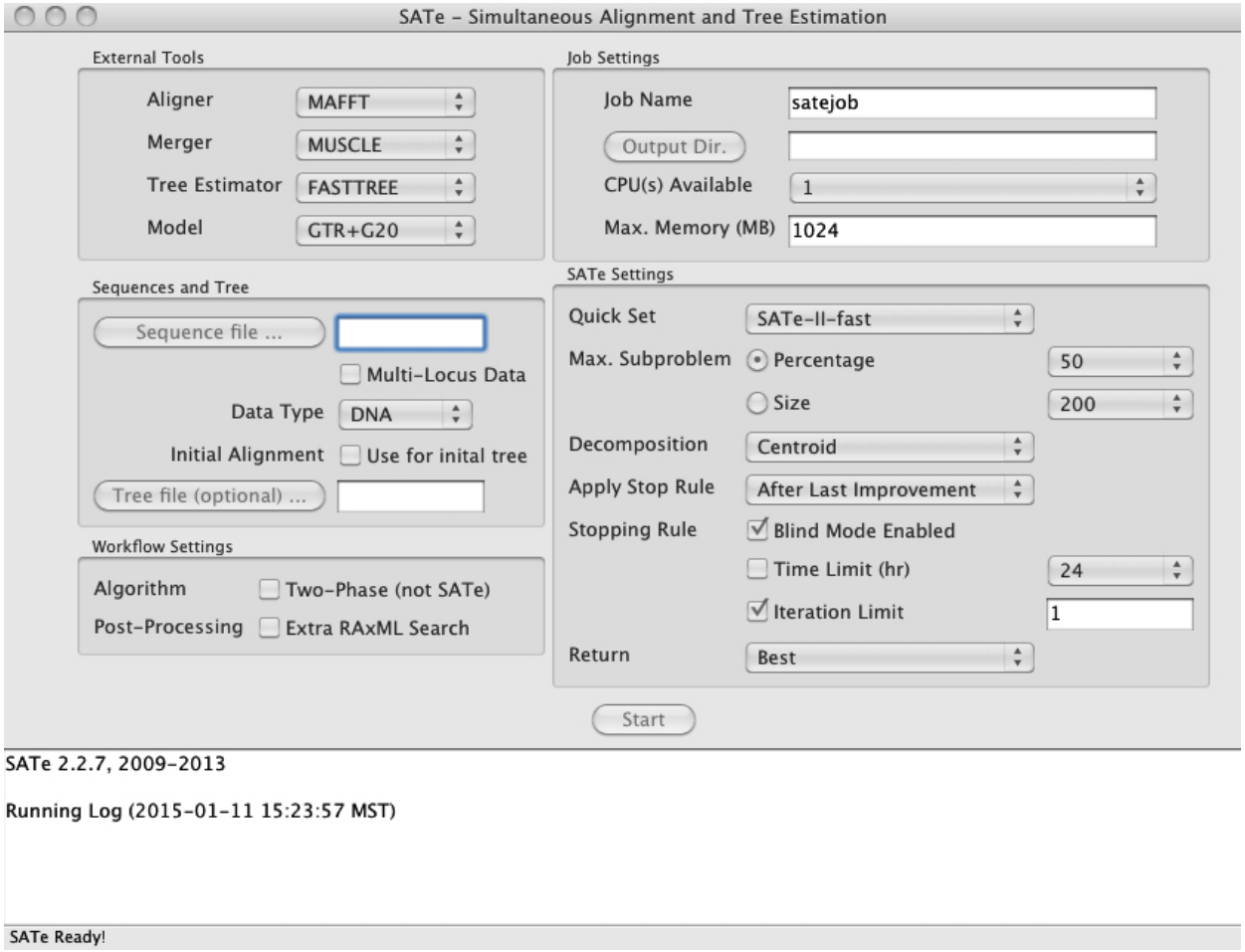
• Create two output folders for the alignment results in your desktop and place the fasta files in the corresponding one

folder: Liolaemus_CYTB
folder: Lizards_RAG1

Running SATe-2

- Open SATe-II GUI by clicking on the executable on the program folder:

- Explore the console and the options in the SATe-II GUI version:



Running SATe-2

• Explore the options in the SATe-II GUI version:

External Tools:
Aligner: [ClustalW2, MAFFT, PRANK, OPAL]
Merger: [MUSCLE, OPAL]
Tree Estimator: [RAXML, FASTTREE]
Model: [RAxML-options: GTRCAT, GTRGAMMA, GTRGAMMAI;
FASTTREE-options: GTR+G20, GTR+CAT, JC+G20, JC+CAT]

Sequences and Tree:
Sequence file ...: [This is the folder where our fasta file resides]
Multi-locus Data [option]
Data Type: [DNA, RNA, Protein]
Initial Aligment [option]
Tree file (optional): [Provide if you have an initial phylogeny associated with the sequences]

Workflow Settings:
Algorithm [option] Two-Phase (not SATe-II)
Post-Processing [option] Extra RAxML Search

Running SATe-2

• Explore the options in the SATe-II GUI version:

Job Settings:
- Job Name: [give a name for the job]
- Output Dir.: [Select the corresponding directory for the output aligment]
- CPU(s) Available: [It will depend on your computer]
- Max. Memory (MB): [It will depend on your computer]

SATe-II Settings
- Quick Set: [Presets: SATe-II_fast, SATe-II_ML, SATe-II_simple, custom]
- Max. Subproblem:
  - Percentage [default 50]
  - Size [default 10]
- Decomposition:
  - Centroid (fast) or Longest (slow)
- Apply Stop Rule: [options]
- Stopping Rule: Blind Mode Enabled
  - Time Limit (hr) [default 24 hours]
  - Iteration limit [default 1 iterations]
- Return: [Default are Final or Best alignment]

Running SATe-2: Select the Following Options

External Tools:
- Aligner: [MAFFT]
- Merger: [MUSCLE]
- Tree Estimator: [RAXML]
- Model: [GTRGAMMAI]

Sequences and Tree:
- Sequence file ...: [Liolaemus_CYTB.fasta]
- Data Type: [DNA]
- Tree file (optional): [None]

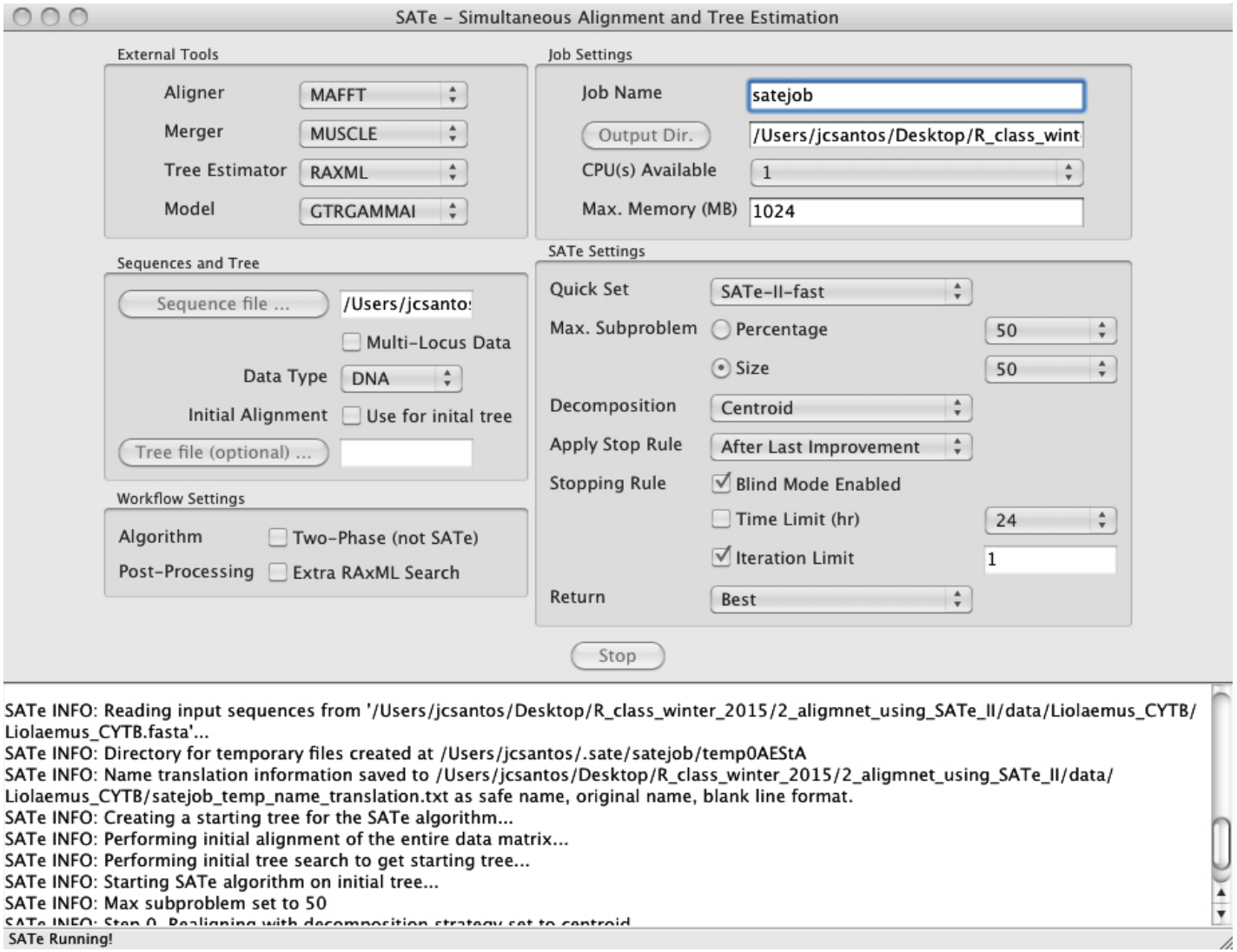
Workflow Settings:
- Algorithm: [None] Two-Phase (not SATe-II)
- Post-Processing: [None] Extra RAxML Search

Job Settings:
- Job Name: [Liolaemus_CYTB_alignment]
- Output Dir.: [Liolaemus_CYTB] Select the corresponding directory for the output alignment
- CPU(s) Available: [2] It will depend on your computer
- Max. Memory (MB): [1000] It will depend on your computer

SATe-II Settings
- Quick Set: [SATe-II_fast]
- Iteration Limit: [3]

Leave other options unchanged

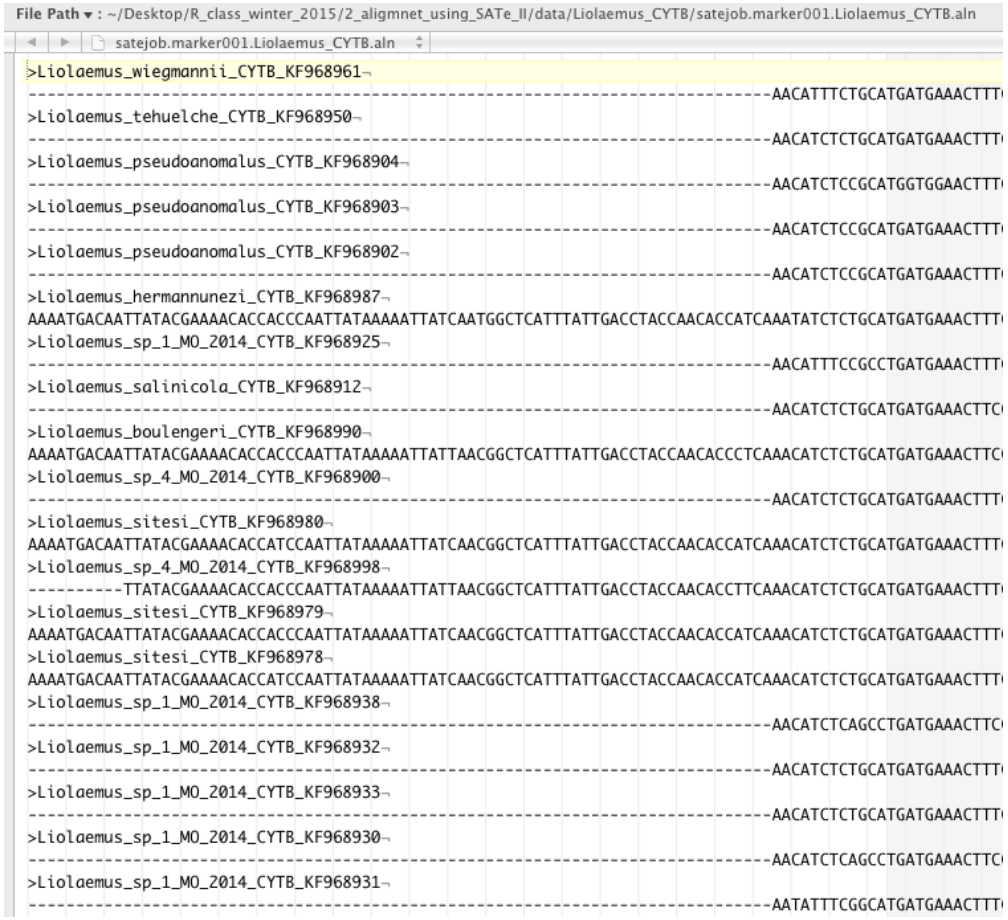
Running SATe-2



Running SATe-2

• Explore the output in a text editor. The alignment is located in these .aln files in fasta format:

satejob.marker001.Liolaemus_CYTB.aln



Repeat the same process with the Lizard_RAG1.fasta file

Mesquite: Visually explore the alignments

- Download mesquite:

http://mesquiteproject.wikispaces.com/Installation+on+MacOS+X

https://github.com/MesquiteProject/MesquiteCore/releases