Species_Tree_Construct_Process

Jialin Yang

```
library(devtools)
#install_github("lliu1871/phybase")
library(phybase)
library(ape)
```

```
genes = c("E", "M", "N", "orf1ab","orf3a","orf6","orf7a","orf7b","orf8","orf10","S")
#genes = c("E", "M", "S")
include_gene = c("E", "M", "N", "orf1ab","orf3a","orf6","orf7a","orf7b","orf8","S")
#include_gene = c('E','S')
ngene = length(genes)

nsim = 100
student_id = 'jy84696'
gacrc_path = '/scratch/jy84696/covid19/RAxML0929/JY/'
local_path = '/Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAx
ML0929/JY/'

nsub = c(1,5,50,100,20,1,2,1,2,1,100)
#nsub = rep(2,times = ngene)

boot_tree_method = c(rep('RAxML',3),'iqtree',rep('RAxML',7))
con_tree_method = c(rep('sumtrees',3),'RAxML',rep('sumtrees',6),'RAxML')
#boot_tree_method = c('RAxML', 'RAxML', 'iqtree')
#con_tree_method = c('sumtrees', 'RAxML', 'RAxML')

#tree_software = 'phylip'
tree_software = 'ape'

con_sptree_method = 'RAxML'
#con_sptree_method = 'sumtrees'
```

Preparation

We first copy the folder “JY” to GACRC.

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/
```

1. Using RAxML v.8.2.12 get best gene trees on GACRC.

For more information about RAxML, type `1besttree/raxmlHPC -h` in terminal.

1.1 Local preparation

```
################################################
# Create 11 Besttree Shells
################################################
set.seed(51129)
seeds = sample(300000:399999, ngene,replace = TRUE)
try(system(paste('scp -r ',local_path,'0alignment/*.reduced ', local_path,'1besttree/
',sep='')))
for (i in 1:ngene){
  sh = paste("#!/bin/sh",
             "#PBS -S /bin/bash",
             " ",
             "#PBS -q batch",
             "#PBS -N RAxML_BestTree",
             "#PBS -l nodes=1:ppn=1",
             "#PBS -l mem=64gb",
             "#PBS -l walltime=168:00:00",
             " ",
             paste("#PBS -M ",student_id,"@uga.edu",sep=""),
             "#PBS -m ae",
             " ",
             paste("cd ", gacrc_path,"1besttree",sep=""),
             "echo 'PBS_JOBID is $PBS_JOBID'",
             " ",
             #"module load RAxML/8.2.11-foss-2019b-pthreads-sse",
             #"ml RAxML/8.2.12-intel-2019b-hybrid-avx2",
             #RAxML/8.2.11-foss-2019b-pthreads-sse.  raxmlHPC-PTHREADS-SSE3
             #RAxML/8.2.12-foss-2019b-hybrid-avx.    raxmlHPC
             #RAxML/8.2.12-foss-2019b-pthreads-avx.  raxmlHPC
             #RAxML/8.2.12-intel-2019b-hybrid-avx2.  raxmlHPC
             " ",
             paste("./raxmlHPC -s ",genes[i], "_al.fasta.reduced -n ", genes[i],
                   "_al.fasta.reduced -m GTRCAT -p",seeds[i], sep=""),
             sep = "\n")
  write(sh,
        file = paste(local_path,'1besttree/subs/besttree_',genes[i],".sh",sep=""))
}
################################################################
# Make an executable file to submit 11 shells at the same time
```

```
#############################################################
qsub = ""
for (i in 1:ngene){
  qsub = paste(qsub, paste("qsub besttree_",genes[i],".sh", sep=""), sep='\n')
}
write(qsub,
      file = paste(local_path,'/1besttree/subs/qsub',sep=""))
try(system(paste(paste('cd ', local_path,'1besttree/subs',sep=""),
                 'chmod +x qsub',
                 sep='\n')))
```

1.2 Submit shells on GACRC.

Copy data(reduced alignment files) and shells to GACRC by running following commands in terminal locally.

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/1besttree/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/1best
tree/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/1besttree/subs/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/
1besttree/subs/
```

Then, login remotely on GACRC, submit shells by running following commands remotely.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/1besttree/subs
./qsub
```

2. Using RAxML v. 8.2.12 (or IQ-TREE v.1.6.12) get 100 bootstrap gene trees for each gene on GACRC.

2.1 Local preparation

Due to the large dataset and long sequences, each gene may have different non-identical sequences and different length. `nsub` is number of shells/jobs you want to run parallel for each gene. As we perform bootstrap support analysis 100 times, `nsub` are 11 integers from 1 to 100 and can be evenly divided by 100.

```
################################################
# Create 11 bootstrap gene trees Shells
################################################
```

```
try(system(paste('scp -r ',local_path,'0alignment/*.reduced ', local_path,'2bootstrap
/',sep='')))

for (i in 1:ngene){
  for (j in 1:nsub[i]){
    # bootstrap gene tree command
    if(boot_tree_method[i]=='RAxML'){
      boot_command = paste("./raxmlHPC -b",sample(300000:399999,1),
                           " -p", sample(300000:399999,1), " -N",nsim/nsub[i]," -m GTRCAT
-s ",
                           genes[i], "_al.fasta.reduced -n ", genes[i],
                           "_al.fasta.reduced_",j,' -k', sep="")
    }else{# use iqtree
      boot_command = paste("module load IQ-TREE/1.6.12-foss-2019b",
                           " ",
                           paste('scp ',genes[i],'_al.fasta.reduced ', genes[i],'_al.
fasta.reduced_',j,sep=''),
                           paste('iqtree -s ', genes[i],'_al.fasta.reduced_',j,' -see
d ',
                                 sample(50000:60000,1),' -bo ',nsim/nsub[i],' -m GTR'
,sep=''),#' --runs 2' run parallel
                           sep = "\n")
    }

    sh = paste("#!/bin/sh",
               "#PBS -S /bin/bash",
               " ",
               "#PBS -q batch",
               "#PBS -N RAxML_Bootstrap",
               "#PBS -l nodes=1:ppn=1",
               "#PBS -l mem=64gb",
               "#PBS -l walltime=168:00:00",
               " ",
               paste("#PBS -M ",student_id,"@uga.edu",sep=""),
               "#PBS -m ae",
               " ",
               paste("cd ", gacrc_path,"2bootstrap",sep=""),
               "echo 'PBS_JOBID is $PBS_JOBID'",
               " ",
               #"module load RAxML/8.2.11-foss-2019b-pthreads-sse",
               #"ml RAxML/8.2.12-intel-2019b-hybrid-avx2",
               " ",
               boot_command,
               sep = "\n")
    write(sh,
          file = paste(local_path,'2bootstrap/subs/boottree_',genes[i],'_',j,".sh",se
p=""))
```

```
  }
}

################################################
# Make an executable file to submit all shells at the same time
################################################
qsub = ""
for (i in 1:ngene){
  for (j in 1:nsub[i]){
    qsub = paste(qsub, paste("qsub boottree_",genes[i],'_',j,".sh", sep=""), sep='\n'
)
  }
}
write(qsub,
      file = paste(local_path,'2bootstrap/subs/qsub',sep=""))

try(system(paste(paste('cd ', local_path,'2bootstrap/subs',sep=""),
                 'chmod +x qsub',
                 sep='\n')))
```

2.2 Submit shells on GACRC.

Copy data(reduced alignment files) and shells to GACRC by running following commands in terminal locally.

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/2bootstrap/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/2boo
tstrap/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/2bootstrap/subs/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY
/2bootstrap/subs/
```

Then, login remotely on GACRC, submit shells by running following commands remotely.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/2bootstrap/subs
./qsub
```

After all jobs are finished, run the following code to combine gene trees for each gene into one single file called [GeneName]_al.fasta.reduced.boottrees.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/2bootstrap/
cat RAxML_bootstrap.E* >> E_al.fasta.reduced.boottrees
cat RAxML_bootstrap.M* >> M_al.fasta.reduced.boottrees
cat RAxML_bootstrap.N* >> N_al.fasta.reduced.boottrees
cat orf1ab*.boottrees >> orf1ab_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf3a* >> orf3a_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf6* >> orf6_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf7a* >> orf7a_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf7b* >> orf7b_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf8* >> orf8_al.fasta.reduced.boottrees
cat RAxML_bootstrap.orf10* >> orf10_al.fasta.reduced.boottrees
cat RAxML_bootstrap.S* >> S_al.fasta.reduced.boottrees
```

```
rm *reduced_*
```

2.3 Build consensus gene trees on GACRC by sumtrees v.4.0.0 (or RAxML)

For small data (our case with 5248 sequences in total), by running the following code, we can get bootstrap gene trees with branch lengths by SumTrees 4.0.0. Consensus gene trees are called *[GeneName]_reduced.con.tre*, where reduced means in the alignment, all identical sequences have been removed. These identical sequences could be added back to the gene trees as a form of polytomy in later sections.

https://dendropy.org/programs/sumtrees.html (https://dendropy.org/programs/sumtrees.html)

For large data (our validation dataset with 40028 human genomes and 50 animals' genomes), Spike and Orf1ab are relatively longer than other genes, thus have more non-identical sequences and harder to get consensus tree in sumtrees. Then, we use RAxML to get the consensus tree WITHOUT branch length. The following command asks RAxML to compute extended majority-rule consensus tree. Output are called *RAxML_MajorityRuleExtendedConsensusTree.[GeneName]_al.fasta.reduced.boottrees.con.tre*, identical sequences will be added back as well in later sections.

Note that in our paper, in order to plot a meaningful consensus gene tree with branch length, RAxML-MRE is used first to make sure all human genomes are in one clade for Spike and Orf1ab, then we only keep one human genome and repeated the method same as other genes, use sumtrees to get a consensus gene tree with branch length.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/2bootstrap/
module load Python/3.9.5-GCCcore-10.3.0
python3 -m pip install git+https://github.com/jeetsukumaran/DendroPy.git
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o E_reduced.con.tre -p E_al.fasta.
reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o M_reduced.con.tre -p M_al.fasta.
reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o N_reduced.con.tre -p N_al.fasta.
reduced.boottrees
./raxmlHPC -mGTRCAT -J MRE -z orf1ab_al.fasta.reduced.boottrees -n orf1ab_al.fasta.re
duced.boottrees.con.tre -p355529
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf3a_reduced.con.tre -p orf3a_a
l.fasta.reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf6_reduced.con.tre -p orf6_al.
fasta.reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf7a_reduced.con.tre -p orf7a_a
l.fasta.reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf7b_reduced.con.tre -p orf7b_a
l.fasta.reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf8_reduced.con.tre -p orf8_al.
fasta.reduced.boottrees
sumtrees.py -F phylip -b 10 --min-clade-freq=0.01 -o orf10_reduced.con.tre -p orf10_a
l.fasta.reduced.boottrees
./raxmlHPC -mGTRCAT -J MRE -z S_al.fasta.reduced.boottrees -n S_al.fasta.reduced.boot
trees.con.tre -p354143
```

2.4 Get short gene trees (with bats, mers, pangolins, and several HUMAN CLADES).

Once consensus gene trees are ready, we need to move the bootstrap gene trees and consensus gene trees from GACRC to our local path.

```
scp -r jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/2bootstrap/*.
reduced.boottrees /Users/jialinyang/Documents/research/data/covid19genetree/my_output
/RAxML0929/JY/2bootstrap/

scp -r jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/2bootstrap/*.
con.tre /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929
/JY/2bootstrap/
```

When use RAxML-MRE(Extended Majority Rule) to do the bootstrap gene trees, it will substitute '|' and '/' by '' *in taxa names. When use SumTrees to build consensus tree, it will substitute "* with " in taxa names. Here we need to specify the bootstrap gene tree method and consensus tree method for each gene, in order to match the taxa names with the original alignment names, and also deal with RAxML consensus tree format.

```
#####################################################################
# Ungroup gene tree: all human + all animals
# add all identical sequences into the consensus gene trees

# Spike use RAxML to build bootstrap trees, and use IQtree to build consensus tree
Ungroup_Gene_Tree = function(treepath, treeinfopath,boot_method,con_method){
  ## input from RAxML results, all '\|/_' are same as alignments
  # treepath: Bootstrap trees folder
  # treeinfopath: identical sequences for bootstrap tree
  # output: ungrouped gene tree and grouped gene tree

  # get all identical sequences pairs
  tree = scan(treepath, what='character')
  tree_info = scan(treeinfopath,  what = "character", sep = '\n')
  tree_info = tree_info[grep("IMPORTANT WARNING: Sequences", tree_info)]
  tree_info = gsub("IMPORTANT WARNING: Sequences ", "", tree_info)
  tree_info = gsub(" are exactly identical", "", tree_info)
  tree_info_df = matrix(NA, nrow = length(tree_info), ncol = 2)
  for (idx in 1:length(tree_info)){
    tree_info_df[idx,] = t(unlist(strsplit(tree_info[idx], split = " and ")))
  }

  # get ungrouped gene tree
  #tree_info_df = gsub('_','',tree_info_df)  # if read con.tre
  if (boot_method=='RAxML'&con_method=='sumtrees'){# for all small dataset
    # if use sumtrees get consensus trees, add this line because sumtrees deleted "_"
    tree_info_df = gsub('_','',tree_info_df)
  }else if(boot_method=='iqtree'&con_method=='RAxML'){# validation data, orf1ab
    # # if use RAxML get bootstrap trees, add this line because RAxML change'/' and '
|'
    tree_info_df = gsub('\\/','_',tree_info_df)
    tree_info_df = gsub('\\|','_',tree_info_df)
  }# boot_method=='RAxML'&con_method=='iqtree' for validation data, Spike, nothing ch
anged
  grpnames = levels(as.factor(tree_info_df[,1]))

  for (name in grpnames){
    grpidx = grep(name, tree_info_df[,1], fixed = T)
    ungrp = paste(paste(tree_info_df[grpidx,2], ':0.0', sep=''),collapse=',')
    ungrp = paste("(",name, ":0.0,", ungrp,")100", sep = "")
    tree = gsub(name, ungrp, tree, fixed = T)
  }
  return(tree)
}

#####################################################################
# Ungroup SHORT gene tree: HUMAN CLADES + all animals
```

```
Ungroup_Short_Gene_Tree = function(tree){
  # tree: ungrouped gene tree string
  # output: ungrouped short gene phylo tree - all animals with human clades
  tree = read.tree(text=tree)
  spname = tree$tip.label
  # find b/p/m non-human index in gene tree
  pangolin_loc = grep("pangolin", spname)
  bat_loc = grep("Rhinolophus",spname)
  bat_loc = c(bat_loc, grep("Hipposideros",spname))
  bat_loc = c(bat_loc, grep("Rousettus",spname))
  bat_loc = c(bat_loc, grep("19/bat",spname))
  bat_loc = c(bat_loc, grep("19_bat",spname))
  bat_loc = c(bat_loc, grep("19\\.bat",spname))
  mers_loc = grep("NC471",spname)
  mers_loc = c(mers_loc,grep("addit",spname))
  loc = sort(c(pangolin_loc,bat_loc,mers_loc))

  drop_idx = c()
  if(loc[1]==1&loc[length(loc)]==length(spname)){
    # clade_size
    clade_size = diff(loc)-1
    clade_size = clade_size[clade_size>0]
    for(j in 1:length(clade_size)){
      # clade start and end location
      clade_start_idx = loc[diff(loc)>1][j]+1
      clade_end_idx = loc[grep(loc[diff(loc)>1][j], loc)[1]+1]-1
      print(paste('From',clade_start_idx,'to', clade_end_idx, 'are human genomes',sep
=' '))
      # keep only one genome for each clade
      tree$tip.label[clade_start_idx] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[j],
sep='')
      if(clade_start_idx<clade_end_idx){
        drop_idx = c(drop_idx, (clade_start_idx+1):clade_end_idx)
      }else{drop_idx = c(drop_idx)}
    }
    tree_short = drop.tip(tree, drop_idx)
  }else if(loc[1]==1&loc[length(loc)]==length(loc)){
    clade_size = length(spname)-length(loc)
    clade_start_idx = length(loc)+1
    clade_end_idx = length(spname)
    print(paste('From',clade_start_idx,'to', clade_end_idx, 'are human genomes',sep='
'))
    # keep only one genome for each clade
    tree$tip.label[clade_start_idx] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[j],se
p='')
    drop_idx = (clade_start_idx+1):clade_end_idx
    tree_short = drop.tip(tree, drop_idx)
```

```
  }else if(loc[1]!=1&loc[length(loc)]!=length(loc)&length(loc)==loc[length(loc)]-loc[
1]+1){
    clade_size = c(loc[1]-1,length(spname)-loc[length(loc)])
    # keep only one genome for each clade
    tree$tip.label[1] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[1],sep='')
    tree$tip.label[loc[length(loc)]+1] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[2]
,sep='')
    drop_idx = c(2:clade_size[1], (loc[length(loc)]+2):length(spname))
    tree_short = drop.tip(tree, drop_idx)
  }else if(loc[1]!=1&loc[length(loc)]==length(spname)){# E
    clade_size = c(loc[1]-1)
    clade_size = c(clade_size, diff(loc)-1)
    clade_size = clade_size[clade_size>0]
    loc = c(0,loc)
    for(j in 1:length(clade_size)){
      # clade start and end location
      clade_start_idx = loc[diff(loc)>1][j]+1
      clade_end_idx = loc[grep(loc[diff(loc)>1][j], loc)[1]+1]-1
      print(paste('From',clade_start_idx,'to', clade_end_idx, 'are human genomes',sep
=' '))
      # keep only one genome for each clade
      tree$tip.label[clade_start_idx] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[j],
sep='')
      if(clade_start_idx<clade_end_idx){
        drop_idx = c(drop_idx, (clade_start_idx+1):clade_end_idx)
      }else{drop_idx = c(drop_idx)}
      tree_short = drop.tip(tree, drop_idx)
    }
  }else if (loc[1]!=1&loc[length(loc)]!=length(spname)){
    loc = c(0,loc,length(spname)+1)
    clade_size = c(loc[1]-1)
    clade_size = c(clade_size, diff(loc)-1)
    clade_size = clade_size[clade_size>0]
    for(j in 1:length(clade_size)){
      # clade start and end location
      clade_start_idx = loc[diff(loc)>1][j]+1
      clade_end_idx = loc[grep(loc[diff(loc)>1][j], loc)[1]+1]-1
      print(paste('From',clade_start_idx,'to', clade_end_idx, 'are human genomes',sep
=' '))
      # keep only one genome for each clade
      tree$tip.label[clade_start_idx] = paste('SARS-COV2-HUMAN-CLADE-',clade_size[j],
sep='')
      if(clade_start_idx<clade_end_idx){
        drop_idx = c(drop_idx, (clade_start_idx+1):clade_end_idx)
      }else{drop_idx = c(drop_idx)}
      tree_short = drop.tip(tree, drop_idx)
    }
```

```
  }

  return(tree_short)
}
```

```
# get short consensus gene tree
for (i in c(1:ngene)){
  if (con_tree_method[i]=='RAxML'){
    # 1.deal with RAxML consensus tree output
    treestring = scan(paste(local_path,'2bootstrap/RAxML_MajorityRuleExtendedConsensu
sTree.',
                            genes[i],'_al.fasta.reduced.boottrees.con.tre',sep=''),
                      what='character',sep=' ')
    # change order of bootstrap values and branch length and remove "[]"
    tree = unlist(strsplit(treestring, '\\)'))
    for (idx in 1:length(tree)){
      bootsupport = substr(tree[idx], unlist(gregexpr('\\[', tree[idx]))[1]+1,unlist(
gregexpr('\\]', tree[idx]))[1]-1)
      branch = substr(tree[idx], 1,unlist(gregexpr('\\[', tree[idx]))[1]-1)
      tree[idx] = substr(tree[idx], unlist(gregexpr('\\]', tree[idx]))[1]+1,nchar(tre
e[idx]))
      tree[idx] = paste(bootsupport,branch,tree[idx],sep='')
    }
    treestring = paste(tree,collapse = ')')
    write(treestring,paste(local_path,'3plot/',
                           genes[i],'_reduced.con.tre',sep=''))
  }else{
    # 2.change sumtree consensus tree output format
    tree = read.tree(paste(local_path,'2bootstrap/',
                           genes[i],'_reduced.con.tre',sep=''))
    tree$tip.label = gsub("'","", tree$tip.label)
    tree$node.label = round(as.numeric(tree$node.label))
    write.tree(tree, paste(local_path,'3plot/',genes[i],'_reduced.con.tre',sep=''))
  }

  # 3. Add identical sequences back into the reduced tree
  treestring = Ungroup_Gene_Tree(paste(local_path,'3plot/', genes[i],'_reduced.con.tr
e',sep=''),
                                 paste(local_path,'0alignment/RAxML_info.',genes[i],'
_al.fasta',sep=''),
                                 boot_method = boot_tree_method[i], con_method = con_
tree_method[i])
  write(treestring, paste(local_path,'3plot/',
                          genes[i],'_0929_full.con.tre',sep=''))
  # 4. Remove most of human genomes, keep only human Clade name and animals
  shorttree = Ungroup_Short_Gene_Tree(treestring)
```

```
  shorttree$tip.label = paste('   ', shorttree$tip.label, sep='')
  shorttree$node.label[which(shorttree$node.label=='NA')]='NaN' # iqtree: spike and o
rf1ab
  # save output
  shorttree_str = write.tree(shorttree, paste(local_path,'3plot/',genes[i],'_0929_sho
rt.con.tre',sep=''))
}
```

3. Plot short gene trees in FigTree or in R. See plots_Rcode folder for R code.

4. Calculate pairwise distance matrices

100 consensus species trees are constructed from average distance matrices. Therefore, for each gene, pairwise distance matrices need to be calculated from 100 bootstrap gene trees. The calculation for 1100 average distance matrices are run parallel at GACRC.

4.1 Save 1100 bootstrap gene trees as input for average distance matrices calculation.

```
##############################################################################
###########
treeinfopath = paste(local_path,'0alignment/RAxML_info.',sep='')

# 1. Ungroup gene trees, because bootstrap gene trees are reduced tree
Ungroup_Gene_Treestr = function(treestr, treeinfopath,boot_method){
  # treepath: Bootstrap trees folder
  # treeinfopath: identical sequences for bootstrap tree
  # output: ungrouped gene tree and grouped gene tree
  # method: RAxML or iqtree(orf1ab)

  # get all identical sequences pairs
  tree = treestr
  tree_info = scan(treeinfopath,  what = "character", sep = '\n')
  tree_info = tree_info[grep("IMPORTANT WARNING: Sequences", tree_info)]
  tree_info = gsub("IMPORTANT WARNING: Sequences ", "", tree_info)
  tree_info = gsub(" are exactly identical", "", tree_info)
  tree_info_df = matrix(NA, nrow = length(tree_info), ncol = 2)
  for (i in 1:length(tree_info)){
    tree_info_df[i,] = t(unlist(strsplit(tree_info[i], split = " and ")))
  }
```

```
  # get ungrouped gene tree
  if (boot_method=='iqtree'){
    tree_info_df = gsub("\\/","_", tree_info_df)
    tree_info_df = gsub("\\|","_", tree_info_df)
  }
  grpnames = levels(as.factor(tree_info_df[,1]))

  for (name in grpnames){
    grpidx = grep(name, tree_info_df[,1], fixed = T)
    ungrp = paste(tree_info_df[grpidx,2], collapse = ",")
    ungrp = paste("(",name, ",", ungrp,")", sep = "")
    tree = gsub(name, ungrp, tree, fixed = T)
  }
  return(tree)
}

for (i in c(1:ngene)){
  # read 100 bootstrap reduced gene trees
  boottrees = scan(paste(local_path,'2bootstrap/',genes[i],'_al.fasta.reduced.boottre
es',sep=''),
                   what = 'character')
  for (bt in 1:nsim){
    # save 100 full boot trees separately
    boottree_j = Ungroup_Gene_Treestr(boottrees[bt],
                                      paste(local_path,'0alignment/RAxML_info.',genes
[i],'_al.fasta',sep=''),
                                      boot_tree_method[i])
    write(boottree_j,
          paste(local_path, "4distmat/data/boot_",bt, "_", genes[i], ".tre", sep = ""
))
  }
}
```

4.2 Prepare 1100 Rcode and GACRC shell scripts for distance matrices calculation

```
# 1. Each gene, each bootstrap, different gene trees.
for (i in 1:ngene){
  for (j in 1:nsim){
    script = scan(paste(local_path,'4distmat/dist_R/dist_mat_Rcode.txt',sep=''),
                  what='character',sep='\n')
    script = c(paste('i = ',i, sep=''),script)
    script = c(paste('bt = ',j,sep=''),script)
    script = c(paste("gacrc_path = '",gacrc_path,"'",sep=''),script)
    script = c(paste("genes=c('",paste(genes,collapse="','"),"')",sep=''),
```

```
                script)
    write(script, paste(local_path, '4distmat/dist_R/distmat_',j,'_',genes[i],'.R',se
p=''))
  }
}

# 2. GACRC shell command to run the R code
for (i in 1:ngene){
  for (bt in 1:nsim){
    sh = paste("#!/bin/sh",
               "#PBS -S /bin/bash",
               " ",
               "#PBS -q batch",
               "#PBS -N dist_mat",
               "#PBS -l nodes=1:ppn=1",
               "#PBS -l mem=64gb",
               "#PBS -l walltime=168:00:00",
               " ",
               paste("#PBS -M ",student_id,"@uga.edu",sep=""),
               "#PBS -m ae",
               " ",

               paste("cd ", gacrc_path,'4distmat/dist_R',sep=''),
               "echo 'PBS_JOBID is $PBS_JOBID'",
               " ",
               "module load R/4.0.0-foss-2019b",
               " ",
               paste("Rscript distmat_",bt,'_',genes[i],".R",sep=""),
               sep="\n")
    write(sh, file = paste(local_path,"4distmat/dist_subs/distmat_",bt,'_',genes[i],"
.sh",sep=""))
  }
}
# 3. An executable file to submit all jobs.
qsub = ""
for (i in c(1:ngene)){
  for (bt in 1:nsim){
    qsub = paste(qsub, paste("qsub distmat_",bt,'_',genes[i],".sh", sep=""), sep='\n'
)
  }
}
write(qsub,
      file = paste(local_path,'4distmat/dist_subs/qsub',sep=""))
try(system(paste(paste('cd ', local_path,'4distmat/dist_subs/',sep=""),
                 'chmod +x qsub',
                 sep='\n')))
```

4.3 Move data, R code and tasks to GACRC

data folder saves all 114$$100 single bootstrap gene trees as well as a file called “spname.txt” with all genome name under study. *dist_R* are the R code to calculate the average distance matrix, this average matrix will be a symmetric matrix with dimension equals number of genomes in “spname.txt”. Note that the average matrix may have missing columns and rows due to gene missing for some genomes, we need to guarantee all of the distance matrices have same dimension with NAs, and then calculate average for non-missing parts.

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/4distmat/data/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/4
distmat/data/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/4distmat/dist_R/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY
/4distmat/dist_R/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/4distmat/dist_subs/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929
/JY/4distmat/dist_subs/
```

Then, login remotely on GACRC, submit task shells by running following commands remotely.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/4distmat/dist_subs
./qsub
```

5. Calculate average distance matrix

Our species tree is based on Neighbor-Joining method, where a distance matrix among all genomes is required. Since we build gene tree first, and then summarize distance matrices for gene trees to get the distance matrix for species, the average distance matrix is calculated from your choice of genes.

In our case, for the small dataset, 5248 genomes are includes, all genes are selected to get an average distance matrix in each bootstrap. However, for validation dataset, 40028 human genomes and 50 animals are included, and orf10 doesn’t have any bats or MERS sequences, so a slightly distance change in orf10 may cause huge bias of the average distance due to large dataset, sometimes the distance of orf10 may lead the average distance among genomes. Therefore, we calculate average distance matrix from all genes except for orf10. You can also specify your own genes into the summarization.

5.1 Prepare nsim=100 Rcode and GACRC shell scripts for distance matrices calculation

```
# 1. Each gene, each bootstrap, different gene trees.
for (bt in 1:nsim){
```

```
  script = scan(paste(local_path,'5outmat/out_R/out_mat_Rcode.txt',sep=''),
                what='character',sep='\n')
  script = c(paste('bt = ',bt,sep=''),script)
  script = c(paste("gacrc_path = '",gacrc_path,"'",sep=''),script)
  script = c(paste("genes=c('", paste(include_gene, collapse="','"),"')",sep=''),
             script)
  write(script, paste(local_path, '5outmat/out_R/outmat_boot',bt,'.R',sep=''))
}

# 2. GACRC shell command to run the R code

for (bt in 1:nsim){
  sh = paste("#!/bin/sh",
             "#PBS -S /bin/bash",
             " ",
             "#PBS -q batch",
             "#PBS -N dist_mat",
             "#PBS -l nodes=1:ppn=1",
             "#PBS -l mem=64gb",
             "#PBS -l walltime=168:00:00",
             " ",
             paste("#PBS -M ",student_id,"@uga.edu",sep=""),
             "#PBS -m ae",
             " ",

             paste("cd ", gacrc_path,'5outmat/out_R',sep=''),
             "echo 'PBS_JOBID is $PBS_JOBID'",
             " ",
             "module load R/4.0.0-foss-2019b",
             " ",
             paste("Rscript outmat_boot",bt,".R",sep=""),
             sep="\n")
  write(sh, file = paste(local_path,"5outmat/out_subs/outmat_boot",bt,".sh",sep=""))
}

# 3. An executable file to submit all jobs.
qsub = ""
for (bt in 1:nsim){
  qsub = paste(qsub, paste("qsub outmat_boot",bt,".sh", sep=""), sep='\n')
}

write(qsub,
      file = paste(local_path,'5outmat/out_subs/qsub',sep=""))
try(system(paste(paste('cd ', local_path,'5outmat/out_subs/',sep=""),
                 'chmod +x qsub',
                 sep='\n')))
```

5.2 Move R code and tasks to GACRC

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/5outmat/out_R/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/5
outmat/out_R/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/5outmat/out_subs/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/J
Y/5outmat/out_subs/
```

Then, login remotely on GACRC, submit task shells by running following commands remotely.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/5outmat/out_subs
./qsub
```

6. Build Neighbor Joining species tree by PHYLIP v.3.697 OR nj function in ape Rlibrary.

nj function in ape library can deal with large distance matrix, thus the species trees are reconstructed from nj function for validation data. PHYLIP is used for small data with 5248 genomes. For convenience, it's better to finish species tree estimation in R along with distance matrices calculation. But the process of using PHYLIP is provided as well.

6.1 Neighbor Joining tree by PHYLIP-neighbor v.3.697

6.1.1 Prepare nsim=100 control files and GACRC shell scripts for neighbor joining tree calculation

```
if(tree_software=='phylip'){
  # 1. Each bootstrap, generate control file
  for (bt in 1:nsim){
    write(paste(paste(gacrc_path, "5outmat/outmat_boot_", bt, sep=""),
                "F",
                paste(gacrc_path, "6outtree/outfile_phylip_", bt, sep=""),
                "Y",
                "F",
                paste(gacrc_path, "6outtree/outtree_phylip_", bt, sep=""),
```

```
                sep = "\n"),
          file = paste(local_path, "6outtree/phylip_control/phylip_control", bt, sep=
""))
  }
  # 2. commands for 100 NJ tree in PHYLIP gacrc subs
  for (bt in 1:nsim){
    sh = paste("#!/bin/sh",
               "#PBS -S /bin/bash",
               " ",
               "#PBS -q batch",
               "#PBS -N NJtree",
               "#PBS -l nodes=1:ppn=1",
               "#PBS -l mem=99gb",
               "#PBS -l walltime=168:00:00",
               " ",
               paste("#PBS -M ",student_id,"@uga.edu",sep=""),
               "#PBS -m ae",
               " ",
               paste("cd ", gacrc_path,'6outtree/phylip_control',sep=''),
               "echo 'PBS_JOBID is $PBS_JOBID'",
               " ",
               "module load PHYLIP/3.697-foss-2019b",

               paste("neighbor < ", gacrc_path,"6outtree/phylip_control/phylip_contro
l", bt,
                     " > ",gacrc_path, "6outtree/phylip_screen/screenout",bt,
                     sep = ""),
               " ",
               sep="\n")
    write(sh,
          file = paste(local_path, "6outtree/phylip_subs/outtree_phylip",bt,".sh",sep
=""))
  }

  # 3. An executable file to submit all jobs.
  qsub = ""
  for (bt in 1:nsim){
    qsub = paste(qsub, paste("qsub outtree_phylip",bt,".sh", sep=""), sep='\n')
  }

  write(qsub,
        file = paste(local_path,'6outtree/phylip_subs/qsub',sep=""))
  try(system(paste(paste('cd ', local_path,'6outtree/phylip_subs/',sep=""),
                   'chmod +x qsub',
                   sep='\n')))
}
```

6.1.2 Move PHYLIP control files and tasks to GACRC.

Then, login remotely on GACRC, submit task shells by running following commands remotely.

6.2 Neighbor Joining tree by R library ape, nj function.

6.2.1 Prepare nsim=100 Rcode files and GACRC shell scripts for neighbor joining tree calculation.

```
if(tree_software=='ape'){
  # 1. Each bootstrap, generate R code file
  for (bt in 1:nsim){
    script = scan(paste(local_path,'6outtree/ape_Rcode/out_tree_Rcode.txt',sep=''),
                  what='character',sep='\n')
    script = c(paste('bt = ',bt,sep=''),script)
    script = c(paste("gacrc_path = '",gacrc_path,"'",sep=''),script)
    write(script, paste(local_path, '6outtree/ape_Rcode/outtree_ape',bt,'.R',sep=''))
  }

  # 2. commands for 100 NJ tree in R gacrc subs
  for (bt in 1:nsim){
    sh = paste("#!/bin/sh",
               "#PBS -S /bin/bash",
               " ",
               "#PBS -q batch",
               "#PBS -N NJtree",
               "#PBS -l nodes=1:ppn=1",
               "#PBS -l mem=99gb",
               "#PBS -l walltime=168:00:00",
               " ",
               paste("#PBS -M ",student_id,"@uga.edu",sep=""),
               "#PBS -m ae",
               " ",
               paste("cd ", gacrc_path,'6outtree/ape_Rcode',sep=''),
               "echo 'PBS_JOBID is $PBS_JOBID'",
               " ",
               "module load R/4.0.0-foss-2019b",
               " ",
               paste("Rscript outtree_ape",bt,".R",sep=""),
               " ",
               sep="\n")
    write(sh,
          file = paste(local_path, "6outtree/ape_subs/outtree_ape",bt,".sh",sep=""))
```

```
  }

  # 3. An executable file to submit all jobs.
  qsub = ""
  for (bt in 1:nsim){
    qsub = paste(qsub, paste("qsub outtree_ape",bt,".sh", sep=""), sep='\n')
  }

  write(qsub,
        file = paste(local_path,'6outtree/ape_subs/qsub',sep=""))
  try(system(paste(paste('cd ', local_path,'6outtree/ape_subs/',sep=""),
                   'chmod +x qsub',
                   sep='\n')))
}
```

6.2.2 Move R files and tasks to GACRC.

```
scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/6outtree/ape_Rcode/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929
/JY/6outtree/ape_Rcode/

scp -r /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/
JY/6outtree/ape_subs/* jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/
JY/6outtree/ape_subs/
```

Then, login remotely on GACRC, submit task shells by running following commands remotely.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/6outtree/ape_subs
./qsub
```

7. Consensus Species Tree estimation

After put all bootstrap specie trees together in a single file, a consensus species tree is constructed by either sumtrees.py OR RAxML-MRE according to number of genomes under study.

Run the following on GACRC.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/6outtree/
awk '{print}' /scratch/jy84696/covid19/RAxML0929/JY/6outtree/outtree_ape_*.txt > /scr
atch/jy84696/covid19/RAxML0929/JY/7contree/outtrees_ape
rm -r /scratch/jy84696/covid19/RAxML0929/JY/6outtree/*.txt
```

Run the following on GACRC if you use sumtrees.py to get the consensns species tree.

Run the following if you use RAxML-MRE to get the consensus species tree.

```
cd /scratch/jy84696/covid19/RAxML0929/JY/7contree
qsub outtrees_ape.sh
```

Copy consensus species tree to local folder.

```
scp -r jy84696@xfer.gacrc.uga.edu:/scratch/jy84696/covid19/RAxML0929/JY/7contree/*.co
n.tre /Users/jialinyang/Documents/research/data/covid19genetree/my_output/RAxML0929/J
Y/7contree/
```

8. Save consensus species tree in newick format.

Save consensus species tree in newick format, called outtrees_[ConsensusSoftware]_abbr.con.tre. where “abbr” means the genome names are replaced indexes by the order in spname.txt file. We will change the abbreviated name back to their original name, and then begin to plot.

```
if (con_sptree_method == 'RAxML'){
  outtree = scan(paste(local_path,'7contree/RAxML_MajorityRuleExtendedConsensusTree.o
uttrees_',tree_software,'.con.tre',sep=''), what='character')
  # change order of bootstrap values and branch length and remove "[]"
  tree = unlist(strsplit(outtree, '\\)'))
  for (idx in 1:length(tree)){
    bootsupport = substr(tree[idx], unlist(gregexpr('\\[', tree[idx]))[1]+1,unlist(gr
egexpr('\\]', tree[idx]))[1]-1)
    branch = substr(tree[idx], 1,unlist(gregexpr('\\[', tree[idx]))[1]-1)
    tree[idx] = substr(tree[idx], unlist(gregexpr('\\]', tree[idx]))[1]+1,nchar(tree[
idx]))
    tree[idx] = paste(bootsupport,branch,tree[idx],sep='')
  }
  outtree = paste(tree,collapse = ')')
  write(outtree, paste(local_path,'7contree/outtrees_',tree_software,'_abbr.con.tre',
sep=''))
}else{# con_sptree_method = 'sumtrees'
  # 2.change sumtree consensus tree output format
    outtree = read.tree(paste(local_path,'7contree/outtrees_',tree_software,'.con.tre
',sep=''))
    outtree$tip.label = gsub("'","", outtree$tip.label)
    outtree$node.label = round(as.numeric(outtree$node.label))
    outtree = write.tree(outtree)
    write(outtree, paste(local_path,'7contree/outtrees_',tree_software,'_abbr.con.tre
',sep=''))
}
```

```
spname = scan(paste(local_path,'4distmat/data/spname.txt',sep=''), what='character')
tree = scan(paste(local_path,'7contree/outtrees_',tree_software,'_abbr.con.tre',sep='
'),what='character')
# change name back
for (i in length(spname):1){
  tree = gsub(paste('Gen',i,sep=''),spname[i],tree)
}
tree = write(tree,paste(local_path,'7contree/outtrees_',tree_software,'_final.con.tre
',sep=''))
```