Community-level Phylogeny Creation

Background

As part of a larger project focused on understanding how land-use change in the *caatinga* region of the Northeast Region of Brazil has altered the composition of pant species in the region, we set out to create a community-level phylogeny so we could analyse the relationships within the community. Due to the rapidly changing nature of the area, the phylogeny will also inform us of species which require collection and genetic sequencing before they are potentially extinct.

Packages, Data and Repeated Functions

Packages:

```
library(tidyverse) # for general data manipulation
library(rentrez) # for accessing the NCBI database
library(pbapply) # provides a timer for the apply family of functions
library(XML) # used to change xml output from NCBI to a list format
library(phylotools) # for importing and altering tree files
```

The data used for creating the phylogeny contains the names and traits of 16,349 angiosperm species found in Brazil. We were only building the phylogeny, so all of the trait data was removed from the data frame to prevent any potential confusion.

Data:

```
(data <- read_csv('nordeste.csv'))

# remove trait data and alter spaces in combination names
(data <- data %>%
  select(Family:Combination) %>%
  mutate(Combination = str_replace_all(Combination, ' ', '_')))
```

A few functions were created to facilitate the creation of the phylogeny.

`seek_seqs` is a rather large function which performs the two vital steps of finding and retrieving sequence data from the NCBI database. This function builds on the `rentrez` package, which already contains the tools for sequence identification and retrieval, by combining the two processes into one and returning the results in an R-friendly format. `seek_seqs` also creates a vector (`seqs`) of the taxa names pulled from the NCBI database for comparison with edited sequence lists using the `del_taxa` function. As the results are of little value in R itself, the `seek_seqs` function also allows for them to be written directly to a .fasta file without saving them to the global environment. If searching for many terms, as is the case with our data, `seek_seqs` can take a considerably long time because NCBI requests that users only query three searches per second.

`write_fasta` builds off of `base::write` to automate the process of writing sequence labels based on provided arguments and save the sequences as a .fasta file.

`del_taxa` uses the `seqs` vector created by `seek_seqs` to compare the sequences obtained from NCBI with the sequence .fasta file following clustering and visual inspection of the generated tree. `del_taxa` returns a vector of sequence names which were removed, allowing for a replacement search to be performed.

Repeated Functions:

```
### seek_seqs ###

seek_seqs <- function(x, GENE = NULL, SLEN = NULL, RET = 1,
                      REPLACEMENT = FALSE, OUTSIDE = FALSE,
                      WRITE = FALSE, ...){

  # x = vector containing taxa names
  # GENE -- a string containing the gene region of interest
  # SLEN -- a string containing the length of desired sequences to be returned
  # RET -- the sequence ID to be returned from the NCBI database;
  #        defaults to the first sequence ID per species per desired gene;
  #        increase if replacing anomalous sequences
  # REPLACEMENT -- whether the search is being used to search
  #        for replacement sequences for deleted taxa or not
  # OUTSIDE -- whether the search is being used to search
  #        for taxa outside of the primary research area
  # WRITE -- whether to write the sequences to a fasta file or not;
  #        see `write_fasta` function for details and arguments
  # ... -- arguments to be passed on to the `write_fasta` function

  ## Error Codes ##

  if(missing(GENE)) stop('GENE argument missing with no default value')

  ## Generate Terms ##

  terms <- paste0(GENE, '[GENE] AND ',
                  x, '[PORG]', # search for primary organism only
                  if(!missing(SLEN)) {paste0(' AND ', SLEN, '[SLEN]')})
  # example term: 'atpB[GENE] AND Hydrocleys_martii[PORG] AND 500:5000[SLEN]'

  ## ID Search ##

  print('Searching for IDs')

  # a function to find and create a vector of IDs from NCBI
  IDs <- unique(unlist(pblapply(terms, function(x){
      tmp <- entrez_search(db = 'nucleotide',
                           term = x,
                           retmax = RET)
      # allow for 0.34 seconds between searches to comply with NCBI's wishes
      Sys.sleep(0.34)

      # create a vector of IDs found from the search of the NCBI database
      if(length(tmp$ids) > (RET - 1)){
      IDs <- tmp$ids[[RET]]}
  })))
  if(length(IDs) == 0) stop('No IDs Found')

  ## Sequence Fetch ##
```

```
  print('Fetching Sequence Data')

  # a function which fetches sequence data in an xml format from NCBI based on
  # ID numbers obtained from the '### ID Search ###' section; results include
  # specimen accession number, taxonomy and sequence, as well as additional data
  seqs <- pblapply(IDs, function(x){
      tmp <- entrez_fetch(db = 'nuccore',
                          id = x,
                          rettype = 'gbc', # download 'complete' data
                          retmode = 'xml') # in an xml format
      Sys.sleep(0.34)

      unlist(tmp, use.names = FALSE)
      # provides an R-friendly list of the xml data retrieved
      tmp <- xmlToList(tmp)

      # replaces the spaces in x with '_' to prevent the sequence label being
      # truncated when creating trees
      rapply(tmp, function(x) str_replace_all(x, ' ', '_'), how = 'replace')
  })

  ## Write Fasta File ##

  # see `write_fasta` function below for details
  if(WRITE == TRUE) {
      print('Writing Fasta File')
      write_fasta(seqs, GENE = GENE, OUTSIDE = OUTSIDE, ...)
  }

  ## Save Vector of Included Taxa ##

  # will save a vector, 'seqs', of combination names to the global
  # environment for comparison with sequence list edited following
  # clustering and visual inspection
  if(REPLACEMENT == FALSE & OUTSIDE == FALSE)
    seqs <<- map_chr(seqs, function(x){
                     str_replace(x$INSDSeq$INSDSeq_organism, '_[[:alpha:]]*\\..*$', '')
  })

  if(REPLACEMENT == FALSE & OUTSIDE == TRUE)
    seqs <<- map_chr(seqs, function(x){
                     str_replace(x$INSDSeq$INSDSeq_organism, '_.*$', '')
  })
}

### write_fasta ###

write_fasta <- function(x, GENE = NULL, ACCESSION = FALSE, FAMILY = FALSE,
                        COMBINATION = FALSE, GENUS = FALSE, COMP = FALSE,
                        OUTSIDE = FALSE, APPEND = FALSE){

  # x -- list of sequence data downloaded from NCBI
  # APPEND -- whether to append the data to an existing file or not;
```

```
  #           'APPEND = TRUE' when adding replacement and outside sequences
  # see below in '### Function ###' for information on other arguments

   ## Error Codes ##

  if(ACCESSION == FALSE & FAMILY == FALSE &
     COMBINATION == FALSE & GENUS == FALSE
     ) stop ('No labels provided for sequences; use ACCESSION,
              FAMILY, COMBINATION and/or GENUS arguments to
              provide labels for sequences')
  if(missing(GENE)
     ) stop ('GENE argument missing with no default value')

  ## Function ##

  fasta <- map_chr(x, function(x){
    paste0('>',
        if(ACCESSION == TRUE) paste0(x$INSDSeq$INSDSeq_locus, '_'),
        if(FAMILY == TRUE) paste0(regmatches(x$INSDSeq$INSDSeq_taxonomy,
                                  regexpr('[[:alpha:]]*aceae',
                                  x$INSDSeq$INSDSeq_taxonomy)), '_'),
        if(COMBINATION == TRUE) str_replace(x$INSDSeq$INSDSeq_organism,
                                            '_[[:alpha:]]*\\..*$', ''),
        if(GENUS == TRUE) str_replace(x$INSDSeq$INSDSeq_organism, '_.*$', ''),
        if(COMP == TRUE) '_comp.', # for composite taxa
        if(OUTSIDE == TRUE) '_OUTSIDE', # for taxa outside the research area
        '\n', x$INSDSeq$INSDSeq_sequence)})

    # writes files to '../results/sequences/GENE/GENE_sequences.fasta'
    # or the Windows equivalent
    write(fasta, file.path('..', 'results', 'sequences', GENE,
                           paste0(GENE, '_sequences.fasta')),
                           append = APPEND)
}

### del_taxa ###

del_taxa <- function(x, OUTSIDE = FALSE) {

  # x -- the gene region of interest used to create the single gene tree;
  #      parameter should be written inside quotations
  # OUTSIDE -- whether or not we or comparing taxa outside of the research area

  if(OUTSIDE == FALSE) {
  tmp <- phylotools::read.fasta(file.path(
         '..', 'results', 'sequences', x,
         paste0(x, '_sequences.fasta')))
  tmp <- str_extract(tmp$seq.name, '[[:alpha:]]*_[^_]*$')
  setdiff(seqs, tmp)
  } else{
    tmp <- phylotools::read.fasta(file.path(
           '..', 'results', 'sequences', x,
           paste0(x, '_sequences.fasta')))
```

```
    tmp <- str_replace(tmp$seq.name, '^.*aceae_', '')
    tmp <- str_replace(tmp, '_.*$', '')
    setdiff(seqs, tmp)
    }
} # outputs contain the taxa removed, allowing for a replacement search
```

Part One - Initial Tree Building

Sequences were downloaded from the NCBI database using the `seek_seqs` function. This function creates search terms based on a vector containing species of interest. A gene region of interest and, if desired, a range of sequence lengths are added to the search term via arguments to the function. An initial search of the NCBI database provides sequence IDs which are then used by the function to download species data from NCBI. The downloaded data contains the taxonomy and sequence, among other data, for the taxa. Finally, `seek_seqs` is used to write the sequence data to a fasta file, with sequence labels in the form of '>Accession_Family_Genus_species', and provides a vector of species' names (`seqs`) for later use.

Initial Accession Search and Sequence Fetch:

```
seek_seqs(data$Combination, GENE = 'atpB', SLEN = '500:5000', RET = 1,
          WRITE = TRUE, ACCESSION = TRUE, FAMILY = TRUE, COMBINATION = TRUE)
# a search of the NCBI database for taxa of interest with sequence data for the
# 'atpB' gene between 500 and 5000 nucleotides in length; writes the retrieved
# data to '../results/sequences/atpB/atpB_sequences.fasta'
```

VSEARCH Clustering

To ensure the sequences downloaded from NCBI were appropriate for creating our trees, we clustered the downloaded sequences around known example sequences. The known sequences were placed at the top of the sequence fasta file and the fasta file was run through a VSEARCH clustering. VSEARCH v2.4.4 is a free and open-source alternative to USEARCH which is useful for large sets of data that would normally require the paid version of USEARCH.

```
vsearch --id 0.40 --notrunclabels --cluster_fast [input] --clusters [output]
```

```
system2('vsearch', args = c('--id 0.40', '--notrunclabels', '--cluster_fast',
        file.path('..', 'results', 'sequences', 'atpB', 'atpB_sequences.fasta'),
        '--clusters', file.path('..', 'results', 'clustering', 'atpB/'))
        # using the `system2()` command lets us directly access the
        # command line from RStudio
```

The argument '–id 0.40' required sequences to be at least 40% similar to the cluster seed (the known sequence) to be part of the cluster. If sequences were less than 40% similar, a new cluster was created. We only used the sequences found in the cluster with at least 40% similarity to the seed.

Mafft Alignment

Alignment was performed with Mafft v7.305b. Gap opening and extension penalties were left at their default values.

```
mafft --adjustdirection --reorder [input] > [output]
```

```
system2('mafft', args = c('--adjustdirection', '--reorder',
        file.path('..', 'results', 'clustering', 'atpB', '0.fasta'), '>',
        file.path('..', 'results', 'alignments', 'atpB', 'atpB_alignment2.fasta'))
```

`[output]` was in the format '/results/alignments/GENE/GENE_alignment.fasta' with 'GENE' being replaced as required in order to ensure the correct use of a later function (`single.tree`). The `--adjustdirection argument` was used to allow mafft to reverse sequences to find the best alignment and `--reorder` was used to change the order of the sequences in the output.

Tree Building with FastTree

Rather than use a pure maximum likelihood method to build the initial trees, we opted for a faster route using the neighbour-joining/approximate-maximum-likelihood method in FastTree v2.1.10. We often would need to build multiple trees based on the same gene and replace a handful of anomalous tips with each iteration; FastTree proved to be a great time saver for this process. Default FastTree parameters were used for a nucleotide alignment (`-nt`).

```
FastTree -nt [input] > [output]
```

```
system2('FastTree', args = c('--nt',
        file.path('..', 'results', 'alignments', 'atpB', 'atpB_alignment.fasta'), '>',
        file.path('..', 'results', 'trees', 'atpB.tree')))
```

Visual Analysis

Trees were visually analysed in FigTree v.1.4.3. Taxa on long branches and anomalous taxa were removed using BioEdit v.7.0.5 with the remaining sequences (gaps removed) overwriting the initial sequence fasta file for comparison with the pre-edited collection of sequences obtained from NCBI.

Replacement:

When possible, the species removed through clustering and visual analysis were replaced by re-running the `seek_seqs` function and only searching for additional instances of the species which were removed. The taxa found in the initial search were saved to the global environment as the value `seqs`, and we used the `del_taxa` function as the input for `seek_seqs` to compare `seqs` with the taxa remaining after clustering and visual inspection, providing a vector of removed taxa.

For example, below we increased `RET` to 2, grabbing the second ID for the desired species where available, ensuring that it was not an ID from the initial search. If we were to run more than one replacement search, we would have simply increased `RET` accordingly. We also added `APPEND = TRUE`, telling the function to add any new sequences to the current sequence file rather than overwriting the file. Similarly, we added `REPLACEMENT = TRUE` to avoid overwriting `seqs` in case further comparison and replacement was required. Note that the vector containing the taxa to be replaced came directly from the `del_taxa` function.

```
seek_seqs(del_taxa('atpB'), GENE = 'atpB', SLEN = '500:5000',
          RET = 2, REPLACEMENT = TRUE, WRITE = TRUE, ACCESSION = TRUE,
          FAMILY = TRUE, COMBINATION = TRUE, APPEND = TRUE)
```

After searching for replacement sequences, we ran **VSEARCH Clustering**, **Mafft Alignment**, **Tree Building with FastTree** and **Visual Analysis** again.

Part Two - Adding Outside Taxa

Following replacement in all single-gene trees, we looked for taxa related to those in our database, but which were outside of the target research area. Specifically, we were interested in genera for which we found no sequences and contained only 1 or 2 species in our research area. To determine the genera which met our requirements, we added the accession numbers of our sequences to our dataframe (`data`) using the `AddAccessionstoDF` function. Once the accession numbers for all gene regions of interest were added to the dataframe, we used the `naFind` function to create a vector of genera which lacked sequence data and contained only 1 or 2 species.

Finding Additional Taxa

```
### add_accessions ###

add_accessions <- function(x, GENE = NULL){

  # x -- data frame containing combination names;
  #      accessions will be added to this data frame
  # GENE -- a string containing the gene region of interest
  #         used to create the single gene tree

  ## Error Codes ##

  if(missing(GENE)) stop('GENE argument missing with no default value')

  ## Function ##

  tmp <- phylotools::read.fasta
         (file.path('..', 'results', 'alignments', GENE,
                    paste0(GENE, '_alignment.fasta')))
  # grabs the accession numbers from the sequence labels
  y <- tibble(str_extract(tmp$seq.name, ('[[:alnum:]]{4,}')))
  # removes everything in the sequence name before the genus,
  # leaving only 'Genus_species'
  y$Combination <- str_replace(tmp$seq.name, '^.*aceae_', '')
  names(y)[1] <- GENE
  join(x, y, by = 'Combination')
}
data <- add_accessions(data, 'atpB')

### na_find ###

na_find <- function(x, geneCount = NULL){

  # x -- dataframe containing accessions for each gene region of interest
  # geneCount -- the number of gene regions of interest for the study

  ## Error Code ##

  if(missing(geneCount)) stop('geneCount argument missing with no default value')

  ## Function ##
```

```
  # sums the presence of elements in the rows of x containing accession numbers
  x$na <- rowSums(!is.na(x[tail(seq_along(x), geneCount)]))
  sum_na <- x %>%
    group_by(Genus) %>%
    summarise(number_of_seqs = sum(na), n = table(Genus))
  na <- sum_na %>%
    filter(number_of_seqs == 0 & n < 3)
  na$Genus
}
```

Below, we were looking to add sequences from outside of the research area so we made `REPLACEMENT = FALSE`, `COMP = TRUE` and `OUTSIDE = TRUE`. We were writing these to the previous sequence fasta file so `WRITE = TRUE` and `APPEND = TRUE`. The sequences were labelled in the format 'Accession_Family_Genus_comp._OUTSIDE' where 'comp.' indicates that the species level is of no interest and we were potentially creating composite taxa for the purpose of concatenating alignments. Note that the vector containing the taxa to be replaced came directly from the `na_find` function. Because of taxonomical synonyms, the genera from the `na_find` function indicated as having no sequence data and fewer than three species could pull sequences with different generic names, possibly even names of genera with plenty of sequences already in our sequence file.

Searching for taxa outside of Brazil:

```
seek_seqs(na_find(data, geneCount = 4), GENE = 'atpB', SLEN = '500:5000',
          RET = 1, REPLACEMENT = FALSE, OUTSIDE = TRUE, WRITE = TRUE,
          ACCESSION = TRUE, FAMILY = TRUE, GENUS = TRUE, COMP = TRUE,
          APPEND = TRUE)
```

VSEARCH Clustering, **Mafft Alignment**, **Tree Building with FastTree** and **Visual Analysis** from **Part One** were run again before searching for any needed replacement sequences for the taxa outside of Brazil. The `del_taxa` function once again provided the input data for `seek_seqs`, this time with the argument `OUTSIDE = TRUE` indicating that we were replacing sequences found outside of the research area. For the `seek_seqs` function, `RET` was once again increased to search for the second (or third, fourth, etc.) instance of a term. `REPLACEMENT = TRUE` was also added to avoid overwriting `seqs`.

Replacement of anomalous taxa outside of Brazil:

```
seek_seqs(del_taxa('atpB', OUTSIDE = TRUE), GENE = 'atpB', SLEN = '500:5000',
          RET = 2, REPLACEMENT = TRUE, OUTSIDE = TRUE, WRITE = TRUE,
          ACCESSION = TRUE, FAMILY = TRUE, GENUS = TRUE, COMP = TRUE,
          APPEND = TRUE)
```

Following replacement of anomalous taxa outside of Brazil, **VSEARCH Clustering**, **Mafft Alignment**, **Tree Building with FastTree** and **Visual Analysis** from **Part One** were run again giving us the final single-gene alignments and trees.

Part Three - Combining Trees

In order to combine the single-gene alignments, we needed to standardise the names of the sequences by removing accession numbers and Mafft alignment directional indicators, leaving only 'Family_Genus_species' and 'Family_Genus_comp._OUTSIDE'. For this, we use the `single_tree` function which removes the accession numbers and mafft directional indicators from the sequence names and saves the sequences to a .fasta file.

Combine Single Gene Trees:

```
single_tree <- function(x) {

  # x -- a string containing the gene used to create the single gene tree;

  tmp <- phylotools::read.fasta(
         file.path('..', 'results', 'alignments', x,
                   paste0(x, '_alignment.fasta')))
  # rewrites 'tmp$seq.name' to remove accession numbers and mafft directional indicators
  tmp$seq.name <- str_extract(tmp$seq.name, ('[[:alpha:]]*aceae_.*$'))
  # rewrites the fasta alignment file
  seqinr::write.fasta(as.list(tmp$seq.text), tmp$seq.name,
                      file.path('..', 'results', 'alignments', x,
                                paste0(x, '_final_alignment.fasta')))
  }
test <- single.tree('atpB')
```

Final Tree Building

Once all of the accession numbers and Mafft directional indicators were removed, Sequence Matrix was used to combine the single-gene alignments and export the combined .phylip file for analysis using RAxML. The final, combined gene tree was built using the RAxML-HPC2 on XSEDE tool on the CIPRES Science Gateway. The tree building analysis was allowed to run for a maximum of 72 hours. All other parameters were left at their default values.