Constructing a phylogenetic tree using R and Python

Beulah Samuel, Nicholas Lizer, Tapiwa Magwaba, Razan Alsayed Omar and Grace Carey

Background

Inventory and DNA-barcode library of ground-dwelling predatory arthropods from Krokar virgin forest, Slovenia

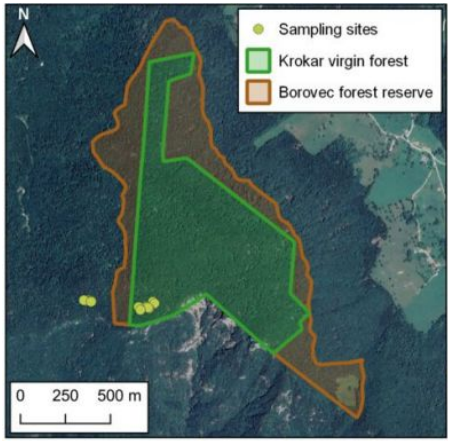
Žan Kuralt‡, Urška Ratajc§, Neža Pajek Arambašić‡, Maja Ferle§, Matic Gabor‡, Ivan Kos‡

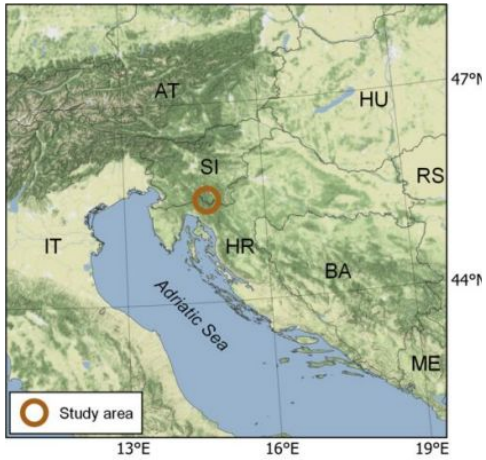
‡ University of Ljubljana, Biotechnical Faculty, Department of Biology, Ljubljana, Slovenia
§ National Institute of Biology, Ljubljana, Slovenia

Presented by Razan

Background

- Location: Krokar forest in Slovenia
  - Virgin forest
  - Most diverse soil invertebrates in Europe
- Lots of information on forest structure but not on ground-dwelling invertebrates
- Importance of studying these invertebrates?
  - Important role in forest soil processes
  - BUT! Climate change: alters habitat
  - Assess change in climate and human impact





Background

- Aims:
  - Generate a checklist of soil and ground-dwelling predatory invertebrates in Krokar
  - Create a DNA-barcode library of these invertebrates
- Taxonomic coverage:

Rank	Scientific Name	Common Name
Order	Araneae	Spider
Class	Chilopoda	Centipedes
Order	Coleoptera	Beetles

https://erwinhuebner.com/seminsects-image-31

Background

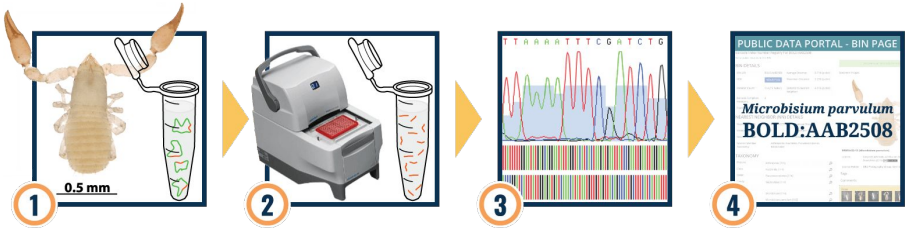
Collected soil samples from Krokar and an adjacent secondary forest

Specimen identification using a stereomicroscope

DNA extraction and sequencing - amplified COI gene

Sequence alignment and phylogenetic tree made using Geneious Prime TreeBuilder

Data deposited on GenBank and BOLD



https://www.universityaffairs.ca/news/news-article/u-of-guelphs-dna-barcode-centre-catalogues-earths-biodiversity/

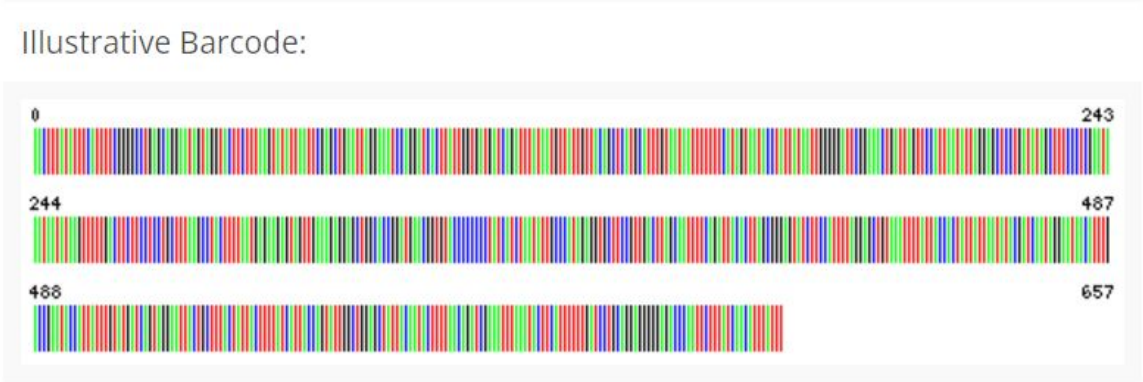
Workflow

Presented by Nic

Column label	Column description
eventID	An identifier of the sampling event, corresponding to the eventID in the "Sampling events" dataset.
order	The name of the order.
scientificName	The full scientific name, with authorship and date information, if known.
sex	The sex of the specimen, if applicable.
taxonRank	The taxonomic rank of the most specific name in the scientificName.
identifiedBy	A list (concatenated and separated) of names of people, groups or organisations who assigned the Taxon to the subject.
dateIdentified	The date on which the subject was identified as representing the Taxon.
basisOfRecord	The specific nature of the data record.
preparations	Type of preservative. Either AP (alcohol preparation) or MP (microscopic slide preparation)
GenBankAccession	GenBank accession code.
occurrenceID	Unique occurrence identifier.
lifeStage	Life stage of specimen. Either adult, subadult or juvenile.
boldSequenceID	Sequence identifier at boldsystems.com

BOLDSYSTEMS (BARCODE OF LIFE DATA SYSTEM)

- Database for barcoded DNA
- example : KROK004-19
  - Points directly to the sample
- COI: Cytochrome c oxidase I
  - Why COI: prevalent, abundant, and distinguishing



1) Data Inspection

- Data formatting
- Visualize initial data

2) Pull sequences

- Decide which accession number is correct.
- Pull sequences using BOLD_id

3) Table rearrangement

- Merge sequence list with original data by process_id column
- Determine columns important for downstream analysis

5) Alignment

- Use MUSCLE to align sequences

6) Distance matrix

- Create distance matrix

7) Build tree

- In Bio.Phylo to create the phylogenetic tree

8) Visualization

- Visualize in python
- Use Matplotlib

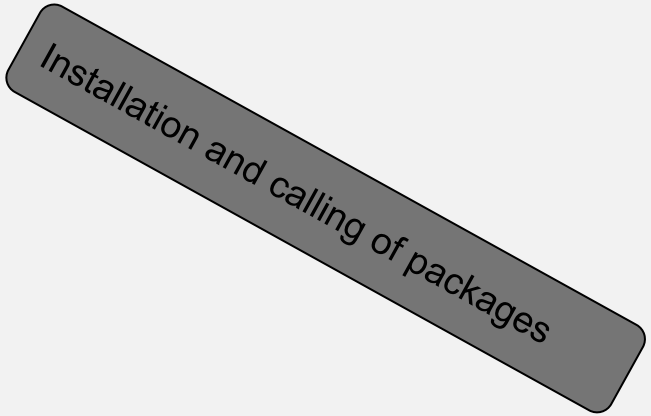
Documentation: Using R

Presented by Tapiwa


ggplot2
R Studio
R Programming
tutorials.jenkov.com

Data inspection

```
```{r,setup, include=FALSE}
library(tidyverse)
library(lemon)
library(knitr)
library(janitor)
library(dplyr )
library(ggplot2)
library(reshape2)
library(bold)
library(spider)
library(data.table)
library(msa)
library(ggtree)
library(treeio)
library(phangorn)
library(seqRFLP)
library(ape)
library(adegenet)
library(ade4)
library(ggimage)
library(TDbook)
#if (!requireNamespace("BiocManager", quietly=TRUE))
#install.packages("BiocManager")
#BiocManager::install("msa")
#if (!require("BiocManager", quietly = TRUE))
    #install.packages("BiocManager")
```
```



```r
RawData <- read.table("DS-KROK4BDJ.txt",
                      header = T,
                      sep = "\t",
                      quote = F, stringsAsFactors = F)
```

```r
dim(RawData)
```

```
[1] 124  80
```

```
kable(column_names)
```

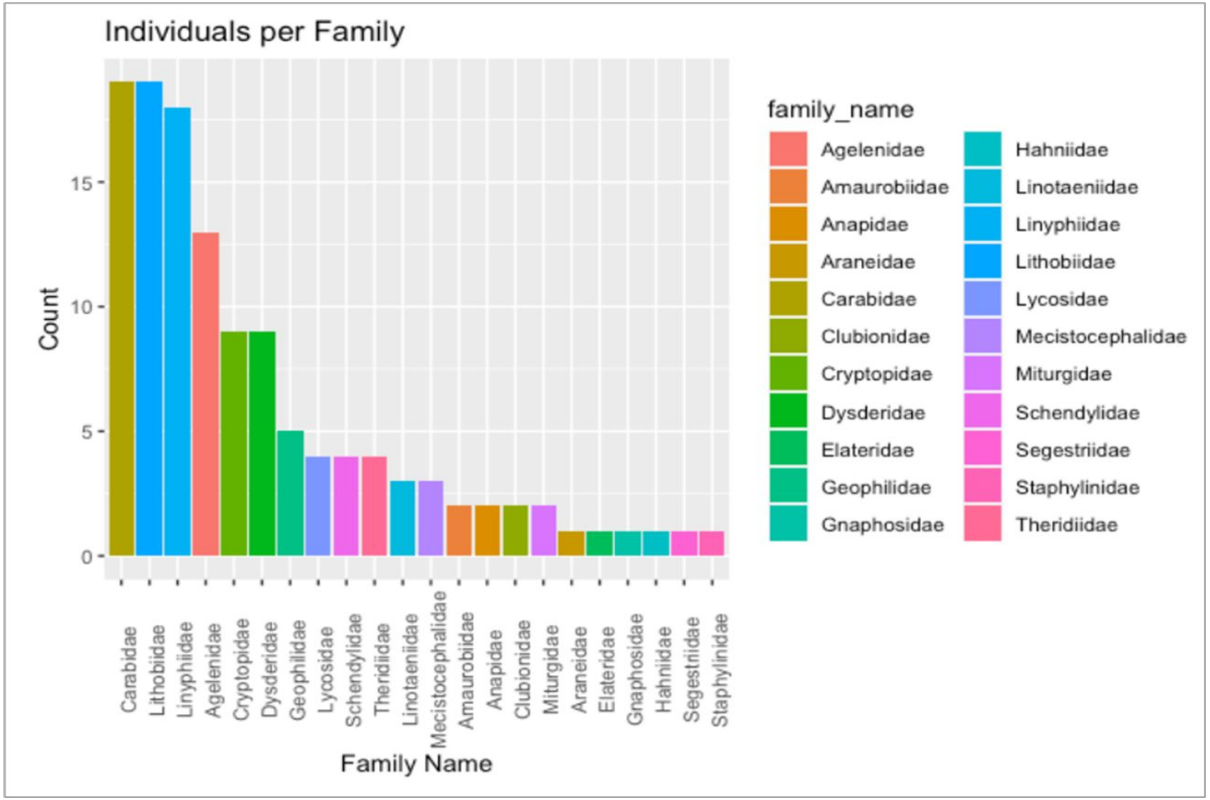
x
processid
sampleid
recordID
catalognum
fieldnum
institution_storing

```
RawData[1:5, 1:10]
```

processid	sampleid	recordID	catalognum	fieldnum
KROK004-19	CLPT-004	10924772	CLPT-004	NA
KROK005-19	CLPT-005	10924773	CLPT-005	NA
KROK008-19	CLPT-015	10924776	CLPT-015	NA
KROK010-19	CLPT-017	10924778	CLPT-017	NA
KROK011-19	CLPT-018	10924779	CLPT-018	NA
KROK012-19	CLPT-019	10924780	CLPT-019	NA
KROK015-19	CLPT-023	10924783	CLPT-023	NA
KROK018-19	CLPT-026	10924786	CLPT-026	NA
KROK019-19	CLPT-027	10924787	CLPT-027	NA
KROK022-19	CLPT-030	10924790	CLPT-030	NA

Visualization

```
ggplot(bdj,
       aes(x=reorder(family_name,
                     family_name,
                     function(x)-length(x)),
      fill=family_name)) +
  geom_bar() +
  labs(title="Individuals per Family",
         x="Family Name",
         y = "Count") +
  theme(axis.text.x = element_text(angle = 90))
```



Retrieve sequences from BOLD

```
RawDataSeq <- bold_seq(taxon = NULL,
                       ids = RawData$processid,   # Unique identifier for the sample
                       bin = RawData$bin_uri,     # Barcode Index Number system identifier
                       container = NULL,
                       institutions = NULL,
                       researchers = NULL,
                       geo = NULL,
                       marker = "COI-5P",         # COI_cytochrome c oxidase I (COI) DNA barcodes
                       response = FALSE)
```

```
str(RawDataSeq)
```

```
List of 115
 $ :List of 4
  ..$ id      : chr "KROK002-19"
  ..$ name    : chr "Abax parallelepipedus"
  ..$ gene    : chr "KROK002-19"
  ..$ sequence: chr "AACATTATACTTTATTTTTGGTGCATGATCAGGAATAGTCGGAACCTCTTTAAGAATGTTAATTCGACTTGAATTAGGAAATCCTGGATCACTAATTGGTGATGACCAAAT"|
 $ :List of 4
  ..$ id      : chr "KROK003-19"
  ..$ name    : chr "Cychrus attenuatus"
  ..$ gene    : chr "KROK003-19"
  ..$ sequence: chr "AACTTTATATTTTATTTTTGGTGCTTGATCAGGGATAGTAGGAACTTCCCTAAGAATACTAATTCGAGCTGAACTAGGAAATCCAGGGTCCTTAATCGGAGATGATCAAAT"|
 $ :List of 4
```

Subset sequences and their IDs

```r
seqlist2<- as.data.frame(matrix(unlist(sequence_df2, use.names = FALSE), ncol = 4, byrow=
TRUE))
```

```r
RawDataSeqReduc <- as.tibble(seqlist2) %>%
  select(1,4)
colnames(RawDataSeqReduc) = c("processid", "retrieved_seq")
```

```r
# covert list to a DF and select seq and IDs
 RawDataSeq_df <- as.data.frame(matrix(unlist(RawDataSeq,
                                              use.names = FALSE),
                                       ncol = 4,
                                       byrow= TRUE)) %>%
  select(1, 4) %>%
```

```r
# convert to a fasta file
RawDataSeq.fasta = dataframe2fas(RawDataSeq_df,
                                 file = "RawDataSeq.fasta.fasta")
```

Alignment using msa package - read in ‘msa’

```
fastatry <- system.file("RawData.fasta", package = "msa")  # file-package customization
mySequences <- readDNAStringSet("RawData.fasta")
mySequences
```

```
DNAStringSet object of length 124:
      width seq                                                                                names
  [1]   658 AACATTATACTTTATTTTTGGTGCATGATCAGGAATAG...AGGAGGTGGAGATCCAATTTTATACCAACATTTATTT KROK002-19
  [2]   658 AACTTTATATTTTATTTTTGGTGCTTGATCAGGGATAG...TGGGGGAGGAGACCCTATTTTATATCAACATTTATTT KROK003-19
  [3]   658 AACTTTATATTTCATTTTCGGGGCCTGAGCAGGAATAG...AGGAGGGGGAGACCCAATTCTTTATCAACATTTATTT KROK004-19
  [4]   658 AACTTTATATTTTATTTTTGGTATATGAGCAGGAATAG...AGGTGGTGGAGACCCTGTTTTATATCAACATTTATTT KROK005-19
  [5]   415 AAATAATATAAGATTTTGACTTCTTCCCCCTTCTTTAA...AGGAGGAGGAGACCCAATTCTCTACCAACATTTATTT KROK006-19
  ...   ... ...
[120]   658 TACTTTGTATTTAATGTTTGGTGCATGATCTGCTATAG...AGGGGGAGGGGATCCTATTTTATTTCAACATTTATTT KROK150-20
[121]   658 TACTTTATATTTAGTTTTTGGAGTTTGATCTGCTATAG...TGGAGGAGGAGATCCTATTTTATTTCAACACTTATTT KROK151-20
[122]   658 TACTTTATATTTGTTGTTTGGTGCTTGATCTGCTATAG...GGGGGGCGGGGATCCAATTTTGTTTCAGCATTTATTT KROK152-20
[123]   658 GACTTTATATTTGATTTTTGGGGTGTGATCAGCTATAG...TGGAGGGGGAGATCCAATTTTATTTCAACATTTATTT KROK153-20
[124]   658 GACTTTATATTTGATTTTTGGGGTGTGATCAGCTATAG...TGGAGGGGGAGATCCAATTTTATTTCAACATTTATTT KROK154-20
> myFirstAlignment <- msa(mySequences)
```

Alignment

```
# Align
myFirstAlignment <- msa(mySequences)
```

Results

Documentation: Using Python

Presented by Beulah

Constructing the phylogenetic tree

Install Biopython using conda and import SeqIO, AlignIO and Phylo

Read Fasta file (generated in “R”) as a list using SeqIO

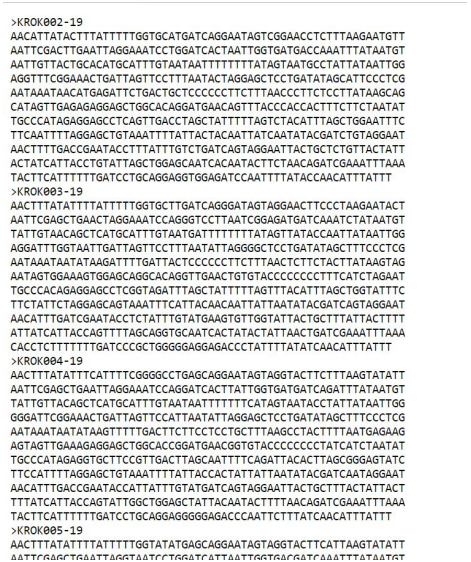
MUSCLE - multiple sequence alignment (CLUSTAL) -> read sequence in python

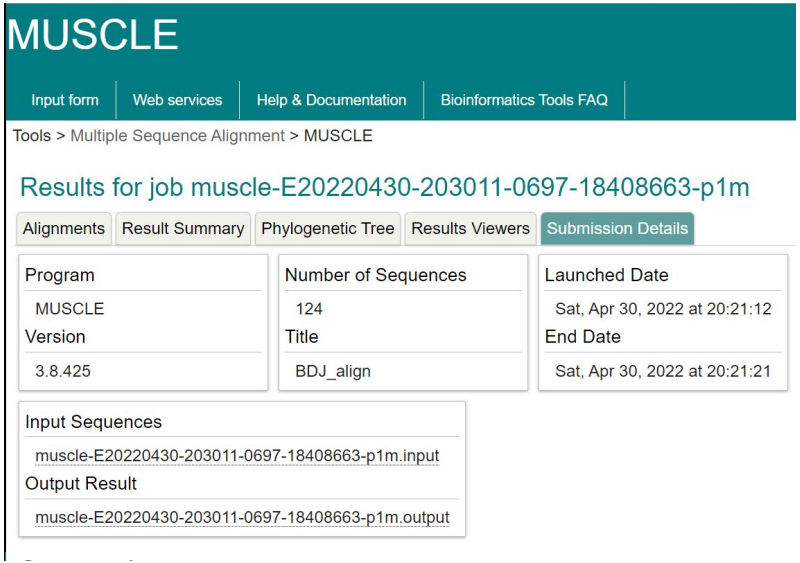
Calculate distance for tree construction

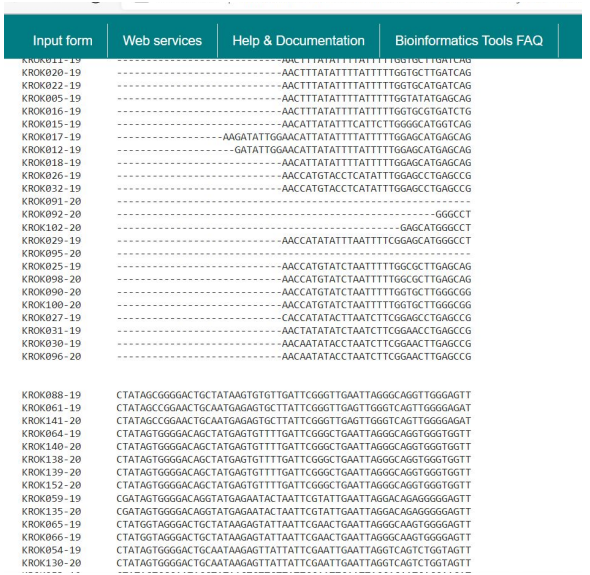
Bio.Phylo.TreeConstruction (neighbour joining)

Using Matplotlib draw and visualize tree

Multiple sequence alignment in MUSCLE







Install Biopython using conda. Import the needed functions from Biopython

In [1]:
```
from Bio import SeqIO
from Bio import AlignIO
from Bio import Phylo
```

fasta file obtained from "R" is read as a list

In [2]:
```
list_fasta = list(SeqIO.parse("df1.fasta", "fasta"))
```

Sequences aligned using MUSCLE online tool. AlignIO function from Biopython is used to read file "bugstuff.aln"

In [3]:
```
with open("bugstuff.aln", "r") as aln:
    alignment = AlignIO.read(aln, "clustal")
```

In [4]:
```
print(type(alignment))
```

```
<class 'Bio.Align.MultipleSeqAlignment'>
```

Use "identity" model to calculate distance for tree construction

In [5]:
```
from Bio.Phylo.TreeConstruction import DistanceCalculator
calculator = DistanceCalculator('identity')
```

Generate the distance matrix

In [6]:
```
distance_matrix = calculator.get_distance(alignment)
print(distance_matrix)
```

```
KROK088-19      0
KROK061-19      0.18410041841004188     0
KROK141-20      0.1827057182705718      0.004184100418409997    0
KROK064-19      0.18967921896792195     0.18967921896792195     0.1882845188284518
KROK140-20      0.1868898186889819      0.1910739191073919      0.1896792189679219
KROK138-20      0.18828451882845187     0.1910739191073919      0.1896792189679219
00139469962     0
KROK139-20      0.18828451882845187     0.1910739191073919      0.1896792189679219
```

Construct tree using "" module in tree construction package

In [7]:
```
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor
constructor = DistanceTreeConstructor(calculator)
```

Use the object created to build the tree

In [8]:
```
bug_tree = constructor.build_tree(alignment)
bug_tree.rooted = True
print(bug_tree)
```

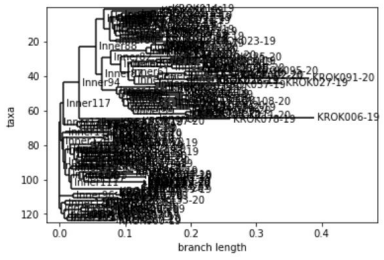
```
Tree(rooted=True)
    Clade(branch_length=0, name='Inner122')
        Clade(branch_length=0.0005457698883205679, name='Inner121')
            Clade(branch_length=0.004631931479173398, name='Inner117')
                Clade(branch_length=0.025079888335387133, name='Inner94')
                    Clade(branch_length=0.025488637108620052, name='Inner88')
                        Clade(branch_length=0.011283357448847744, name='Inner86
```

Write file in "XML" format

In [9]:
```
Phylo.write(bug_tree, "bug_tree.xml", "phyloxml")
```
Out[9]: 1

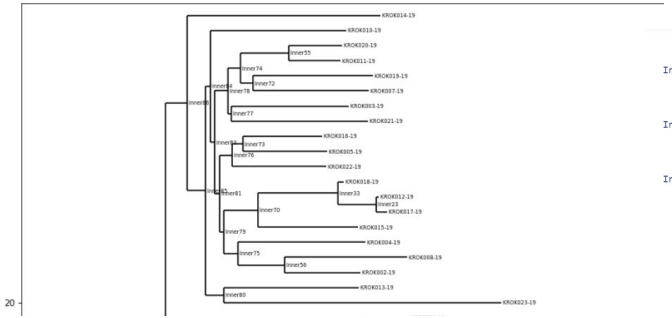
Need to import and work in "matplotlib" to draw the tree

In [10]:
```
import matplotlib
import matplotlib.pyplot as plt
fig = Phylo.draw(bug_tree)
```



Need to reduce overlap of branches

In [11]:
```
fig = plt.figure(figsize=(20, 40), dpi=75) # create figure & set the size
matplotlib.rc('font', size=6)              # fontsize of the leaf and node labels
matplotlib.rc('xtick', labelsize=10)       # fontsize of the tick labels
matplotlib.rc('ytick', labelsize=10)       # fontsize of the tick labels
#bug_tree.ladderize()
axes = fig.add_subplot(1, 1, 1)
Phylo.draw(bug_tree, axes=axes)
fig.savefig("bug_cladogram")
```



Convert() and write() as a "nexus" file

In [12]:
```
Phylo.convert("bug_tree.xml", "phyloxml", "bug_tree.nex", "nexus")
```
Out[12]: 1

In [13]:
```
Phylo.write(bug_tree, "bug_tree.nex", "nexus")
```
Out[13]: 1

In [14]:
```
bug_nex = Phylo.read("bug_tree.nex", "nexus")

fig = plt.figure(figsize=(20, 40), dpi=75) # create figure & set the size
matplotlib.rc('font', size=6)              # fontsize of the leaf and node labels
matplotlib.rc('xtick', labelsize=10)       # fontsize of the tick labels
matplotlib.rc('ytick', labelsize=10)       # fontsize of the tick labels
#bug_tree.ladderize()
axes = fig.add_subplot(1, 1, 1)
Phylo.draw(bug_nex, axes=axes)
fig.savefig("bug2_cladogram")
```

Results

Presented by Grace

Pros and Cons of Phylogenetic tree construction methods

Method	Advantage	Disadvantage	Other information
Maximum parsimony	Appropriate for very similar sequences and a small number of sequences	Very time-consuming as it tests all possible trees Parsimony may fail for diverged sequences Suffers from the long-branch attraction	Predict the evolutionary tree that minimizes the number of steps required to generate the observed variation in the sequences It is built with the fewest changes required to explain (tree) the differences observed in the data
Maximum likelihood	Suitable for very dissimilar sequences We can formulate hypothesis about evolutionary relationships More accurate phylogenetic trees can be constructed for a small number of taxa in a reasonable time frame	A slow search algorithm will lead to slow response Takes a long time for large datasets	It tries to find a model that has the highest probability to generate the input sequence under a given evolutionary model
Neighbour joining	Faster than the character-based method They are fast and can be used with a variety of models	Conversion from sequence data to distance data leads to loss of information	Provides an unrooted tree and a single resultant tree
UPGMA Fitch Mangrolish	Reliable for related sequences Less sensitive to variations in evolutionary rate	Evolution rate is constant in all branches Dependent on the model used to obtain the distance matrix	UPGMA provides rooted tree

Munjal, G., Hanmandlu, M., & Srivastava, S. (2019). Phylogenetics algorithms and applications. In *Ambient Communications and Computer Systems* (pp. 187-194). Springer, Singapore.

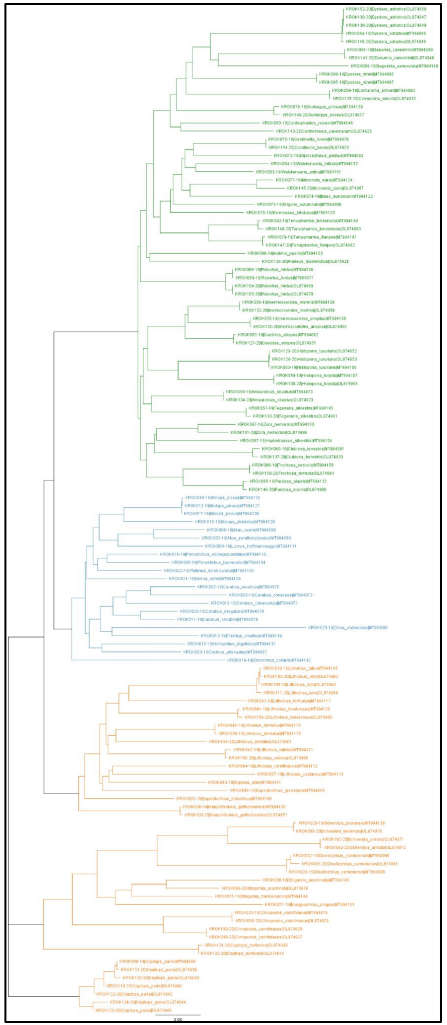
We are attempting to reproduce the author's phylogenetic tree

Published tree:

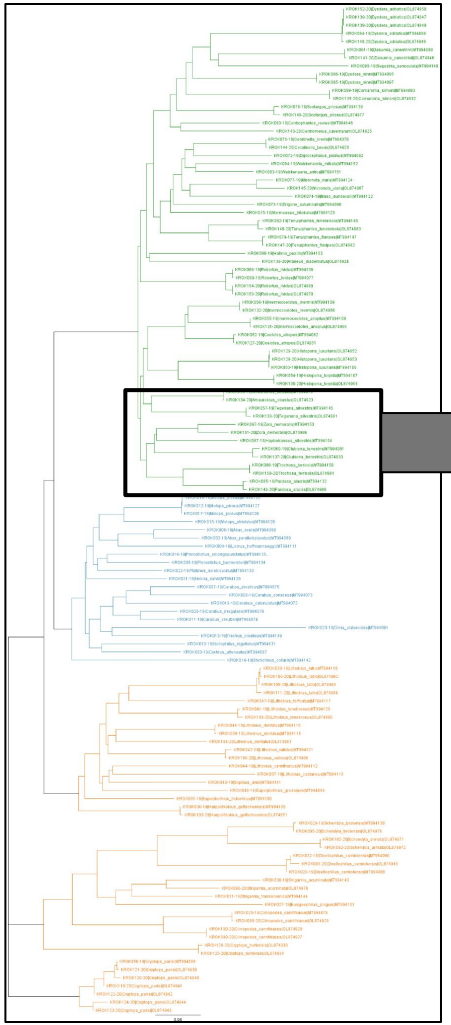
- Built a COI tree using Geneious Prime Tree Builder (Geneious version 2022.0 created by Biomatters)
- Distance matrix was calculated using Global alignment with free end gaps and 70% similarity (IUB)(5.0/-4.5) cost matrix
- The tree was built with Tamura-Nei genetic distance and the Neighbour-Joining tree build method.

Our tree:

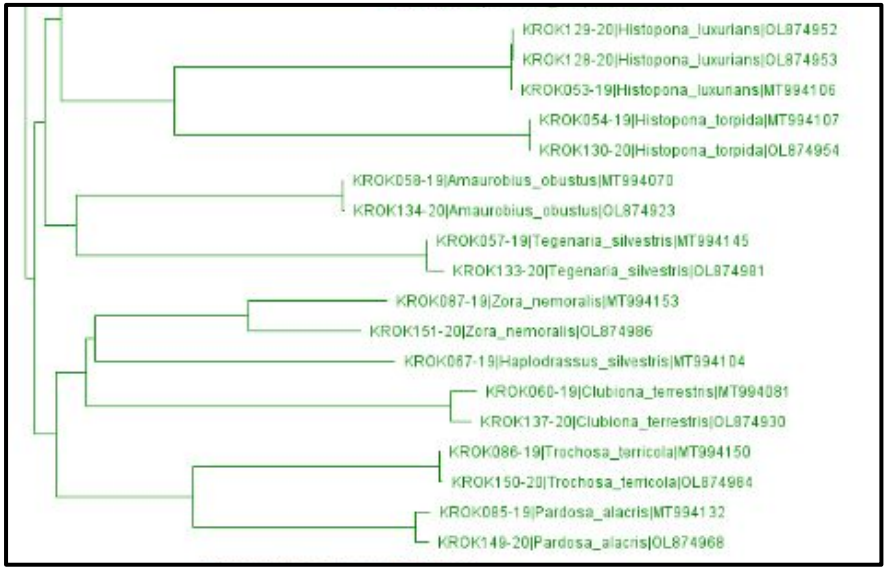
- Used distance- and alignment-based methods to calculate the phylogenetic tree
- Followed the author's specifications using R and BioPython
- **Our goal: transparency and reproducibility**
- Files stored in github repository "https://github.com/gcarey1/BCB_final_collab



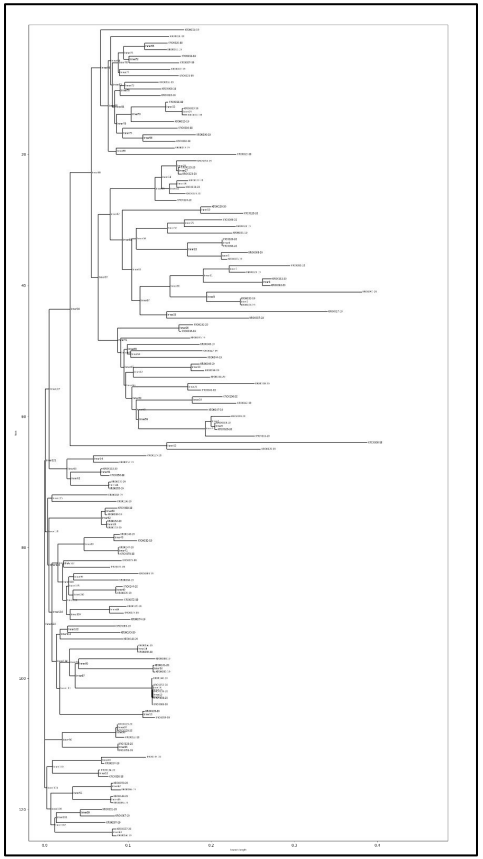
Original Phylogenetic tree



Original Phylogenetic tree

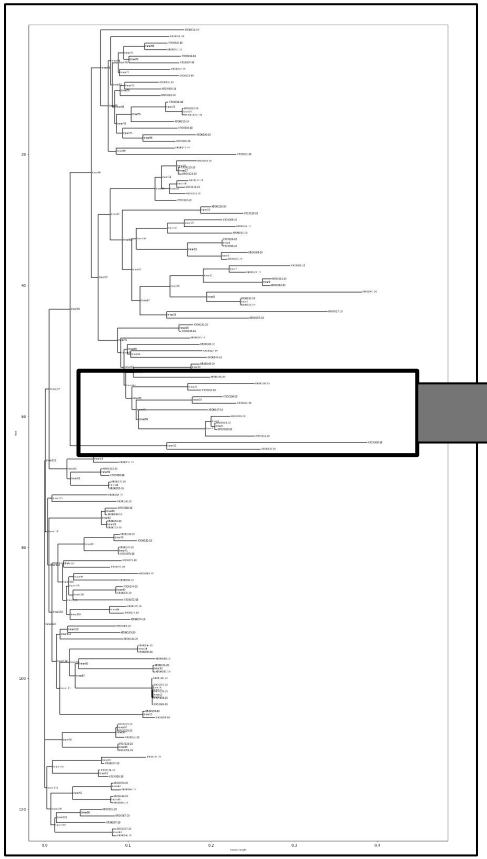


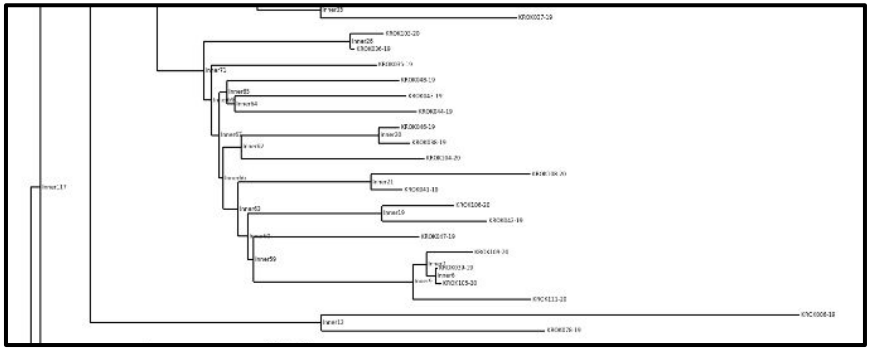
Rank	Scientific Name	Common Name
Order	Araneae	Spider
Class	Chilopoda	Centipedes
Order	Coleoptera	Beetles

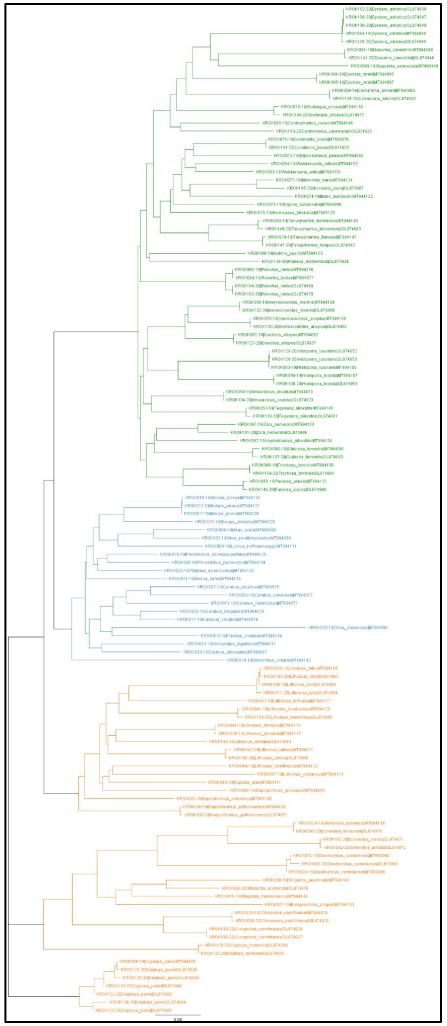


BioPython Phylogenetic Tree

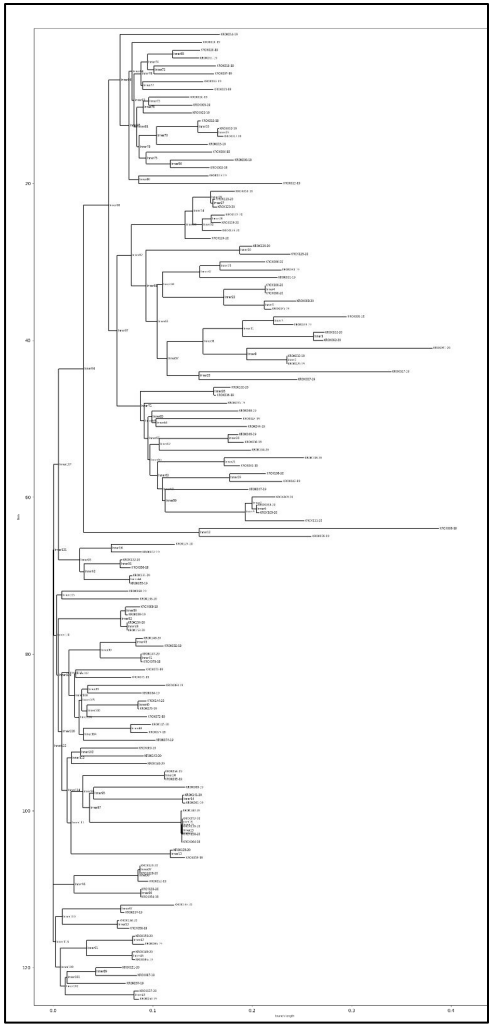
BioPython Phylogenetic Tree



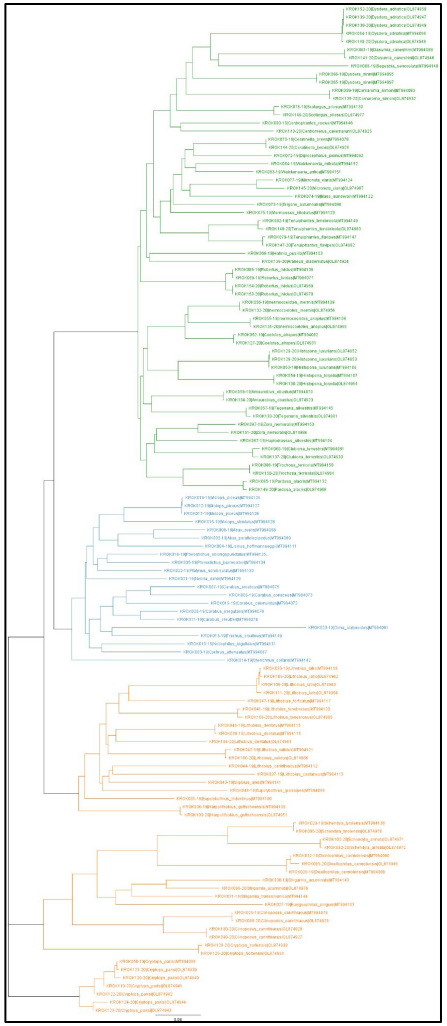




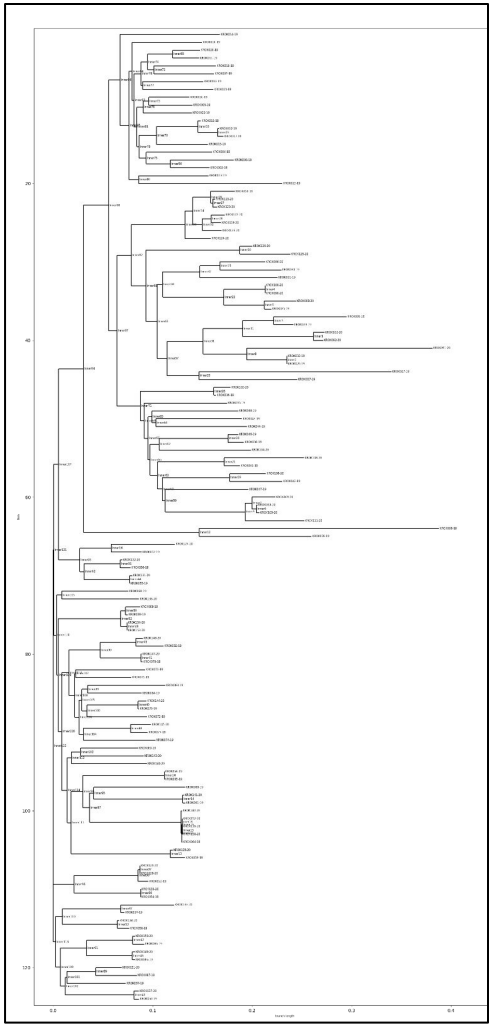
VS



Side-by-side comparison

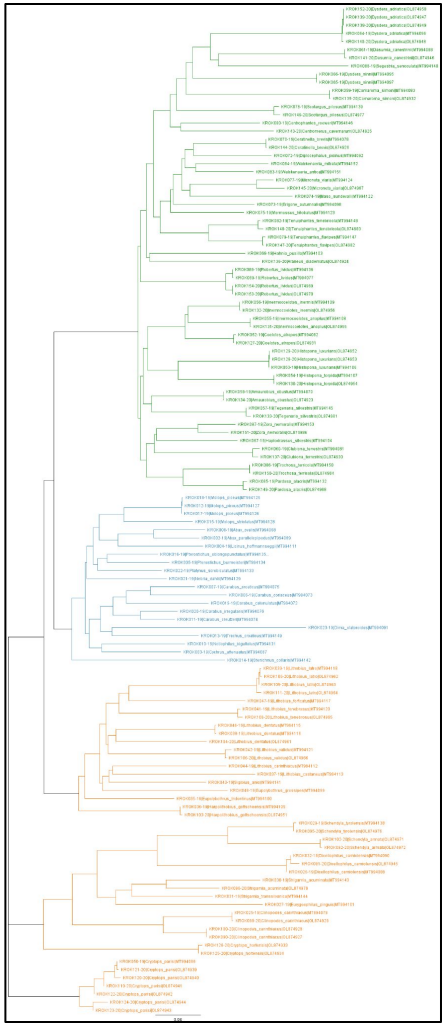


VS

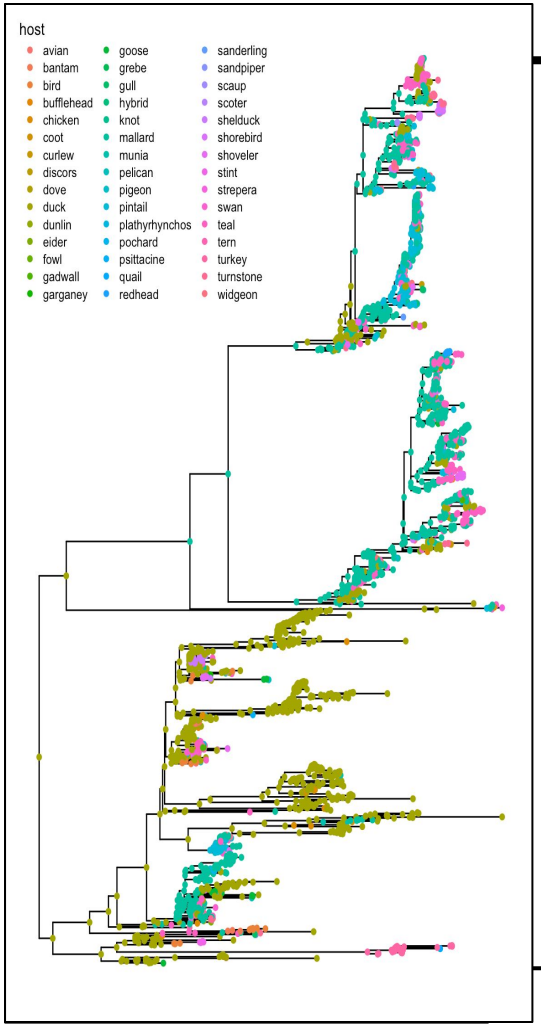


Points to improve:

- Color coding
- Species/subspecies designation instead of BOLD ID
- Wider spacing of nodes
- Classify by order (not class/class/order)
- Ideal visualization package: ggtree (R)



VS



Points to improve:

- Color coding
- Species/subspecies designation instead of BOLD ID
- Wider spacing of nodes
- Classify by order (not class/class/order)
- Ideal visualization package: ggtree (R)

Reproducibility is the goal

https://guangchuangyu.github.io/ggtree-book/chapter-ggtree.html