phruta: scrapping genbank and assembling phylogenetic trees [*]

Cristian Román-Palacios *School of Information, University of Arizona, Tucson, Arizona 85721, USA.*
ORCiD : 0000-0003-1696-4886

Current methodological practices for assembling phylogenetic trees often recur to sequence data stored in GenBank. However, understanding molecular and taxonomic availability in GenBank is generally not very straightforward. For instance, the genetic makeup of datasets available in GenBank can strongly differ between genera even within the same family. Similarly, the taxonomic information associated with sequence data in GenBank can be outdated, relative to other databases that mainly focus on the taxonomic side. phruta, a newly developed R package, is designed to improve the user experience and access to information to genetic data stored in GenBank. By using phruta, users are able to (1) quantitatively explore the molecular makeup of particular clades with information in GenBank, (2) assemble curated multi-gene molecular datasets with retrieved and local sequences, and (3) run basic phylogenetic talks, all within R. The structure of the functions implemented in phruta, designed as a workflow, aim to allow users to assemble simple workflows for particular talks, which are in turn expected to increase reproducibility when assembling phylogenies. This paper provides a brief overview on the performance and workflow associated with phruta.

Keywords: R package, Phylogenetics, Reproducibility, Workflow

Background

The `phruta` `R` package

The `phruta` package is designed to simplify the basic phylogenetic pipeline in `R`. `phruta` is expected to allow scientists from different backgrounds to assemble molecular databases or phylogenies for particular taxa with as minimal complexity and maximal reproducibility as possible. All the code in `phruta` is run within the same software (`R`) and data from intermediate steps are either stored to the environment or can be exported locally to different folders. In general, `phruta` is able to (1) find potentially (phylogenetically) relevant gene regions for a given set of taxa based on GenBank, (2) retrieve gene sequences and curate taxonomic information from the same database, (3) combine downloaded and local gene sequences, and (4) perform sequence alignment, phylogenetic inference, and basic tree dating tasks.

[*]Replication files are available on the author's Github account (http://github.com/cromanpa). **Current version**: September 02, 2022; **Corresponding author**: cromanpa94@arizona.edu.

Alternatives to phruta

Similar functionalities for assembling curated molecular datasets for phylogenetic analyses can be found in `phylotaR` and SuperCRUNCH. However, `phylotaR` is limited to downloading and curating sequences (e.g. it does not align sequences). Similarly, `SuperCRUNCH` only curates sequences that are already stored locally. In fact, `phruta` is closer to `SUPERSMART` and the associated `R` workflow `SUPERSMARTR`. However, most of the applications in the different packages that are part of `SUPERSMARTR` are simplified in `phruta`. Standalone applications that might resemble `phruta` could include `MEGA` and geneious. However, analyses in these two alternatives are either poorly reproducible (e.g. `MEGA`) or not all the functions are freely available to everyone (e.g. geneious has a paid version).

`phruta` in a nutshell

The current release of `phruta` includes a set of eight major functions. Running all eight major functions in `phruta` results in a time-calibrated phylogeny. However, users interested in using their own files at any stage can run each function independently. Note that all the functions for which their primary output are sequences (aligned or unaligned) are listed under `sq.*`. All the files that output phylogenies (time-calibrated or not) are listed under `tree.*`.

- First, the distribution of gene sampled for a given organism or set of taxa can be explored using the `acc.gene.sampling` function. This function will return a table that summarizes either the distribution of genes sampled for the search term in general or specifically across species.

- Second, given a list of target organisms, users can retrieve a list of accession numbers that are relevant to their search using `acc.table.retrieve()`. Instead of directly downloading sequences from genbank (see `sq.retrieve.direct()` below), retrieving accession numbers allow users to have more control over the sequences that are being used in the analyses. Note that users can also curate the content of the dataset obtained using `sq.retrieve.direct()`.

- Third, users should download gene sequences. Sequences can be download using the `sq.retrieve.indirec` from the accession numbers retrieved before using the `acc.table.retrieve()` function.

This is the preferred option within `phruta`. Additionally, users can directly download gene sequences using the `sq.retrieve.direct()` function. Both `sq.retrieve.indirect()` and `sq.retrieve.direct()` functions save gene sequences in `fasta` files that will be located in a new directory named `0.Sequences`.

- Fourth, `sq.add()` allows users to include local sequences to those retrieved from genbank in the previous step. This function saves all the resulting `fasta` files in two directories, combined sequences in `0.Sequences` and local sequences in `0.AdditionalSequences` (originally downloaded sequences are moved to `0.0.OriginalDownloaded` at this step). Note that `sq.add()` is optional.
- Fifth, the `sq.curate()` function filters out unreliable sequences based on information listed in genbank (e.g. `PREDICTED`) and on taxonomic information provided by the user. Specifically, this function retrieves taxonomic information from the Global Biodiversity Information Facility (GBIF) database's taxonomic backbone (see alternatives in the advanced vignette to `phruta`). If a given species belongs to a non-target group, this species is dropped from the analyses. This function automatically corrects taxonomy and renames sequences.
- Sixth, `sq.aln()` performs multiple sequence alignment on `fasta` files. Currently, `phruta` uses the `DECIPHER` R package, here. This package allows for adjusting sequence orientation and masking (removing ambiguous sites).
- Seventh, the `tree.raxml()` function allows users to perform tree inference under `RAxML` for sequences in a given folder. This is a wrapper to `ips::raxml()` and each of the arguments can be customized. The current release of `phruta` can manage both partitioned and unpartitioned analyses. Starting and constrained trees are allowed.
- Eight, `tree.dating()` enables users to perform time-calibrations of a given phylogeny using `geiger::congruify.phylo()`. `phruta` includes a basic set of comprehensively sampled, time-calibrated phylogenies that are used to extract secondary calibrations for the target phylogeny. Note that sampling in those phylogenies can be examined using `data(SW.phruta)`. Please make sure you have at least **two** groups in common with each of the phylogenies. Similarly, users can choose to run either `PATHd-8` or `treePL`.

Assembling a molecular dataset for target taxa in phruta

Let's learn how `phruta` works by assembling a molecular dataset at the species level for a few bird clades. For this tutorial, we will focus on assembling a phylogeny for the new world Quails (family Odontophoridae), with nearly 34 extant species classified in 10 genera (@ref(fig:quail)). We will use the Phasianidae as an outgroup. Within this clade, we will particularly focus on the genus **Polyplectron** with 8 extant species. Luckily, we will be able to compare how similar is the resulting tree for the Odontophoridae relative to pushed phylogenies (Crowe et al. 2006a, b; Cohen et al. 2012; Hosner et al. 2015). You can explore the functionalities of `phruta` using other taxonomic groups of your choice.

```
library(phruta)
```

So far, we have decided the taxonomic makeup of our analyses. From this point, we could simply check the genetic sampling of previous studies and search for those genes in GenBank for the target taxa (Crowe et al. 2006a, b; Cohen et al. 2012; Hosner et al. 2015). For instance, [review sampling in each of those]. We could use these gene names to assemble a molecular dataset for the Odontophoridae and **Polyplectron** in `phruta`. Alternatively, we could use `phruta` to figure out what genes are well sampled in GenBank for both the ingroup and outgroup. We will do the latter in this paper. For this, we will use the `gene.sampling.retrieve()` function in `phruta`. The resulting `data.frame`, named `gs.seqs` in this guide, will contain the list of full names for genes sampled in GenBank for the target taxa.

```
gs.seqs <- gene.sampling.retrieve(organism = c("Odontophoridae", "Polyplectron"),
                                  speciesSampling = TRUE)
```

For the search terms, `phruta` was able to retrieve the names for 79 gene regions from GenBank. The frequency estimates per gene are based on inter-specific sampling (@ref(tab:topGenes)). Note that the `gene.sampling.retrieve()` function provides an estimate of the number of species in GenBank that match the taxonomic criteria of the search term and that have sequences for a given gene region. However, this estimate is only as good as the annotations for genes deposited in GenBank.

From this point, we will generate a preliminary summary of the accession numbers retrieved for the combination of target taxa and gene regions. I call it preliminary because not all these accession numbers are expected to be in the final molecular dataset. For instance, some sequences could be labeled in GenBank under different species, which turn out being synonyms.

We will now assemble a species-level summary of accession numbers using the `acc.table.retrieve()` function (i.e. `speciesLevel = TRUE` argument). For simplicity, this tutorial will focus on analyzing gene regions that are sampled in >20% of the species (`targetGenes` data.frame). The `acc.table` object created below is a `data.frame` object that will be later used to download the relevant gene sequences from GenBank (@ref(tab:AccN)).

```r
targetGenes <- gs.seqs[gs.seqs$PercentOfSampledSpecies > 20,]
acc.table <- acc.table.retrieve(
            clades  = "Odontophoridae",
            species = "Polyplectron",
            genes = targetGenes$Gene,
            speciesLevel = TRUE
          )
```

Since we're going to retrieve sequences from GenBank using an existing preliminary accession numbers table, we will use the `sq.retrieve.indirect()` function in `phruta`. Please note that there are two versions of `sq.retrieve.*` in `phruta` . The one that we're using in this guide, `sq.retrieve.indirect()`, retrieves sequences "indirectly" because it requieres a table of accession numbers (see the `acc.table.retrieve()` function above). I present the information in this vignette using `sq.retrieve.indirect()` instead of `sq.retrieve.direct()` because `sq.retrieve.indirect()` is more flexible. Specifically, `sq.retrieve.indirect()` allows for correcting issues *prior* to downloading/retrieving the sequences. For instance, you can add new sequences, species, populations to the resulting `data.frame` from `acc.table.retrieve()`. Additionally, you could even manually assemble your own dataset of accession numbers to be retrieved using `sq.retrieve.indirect()`. Instead, `sq.retrieve.direct()` does its best to directly retrieve GenBank sequences for a target set of taxa and set of gene regions. In short, you should be able to catch errors using `sq.retrieve.indirect()` but mistakes will be harder to spot and fix if you're using `sq.retrieve.direct()`.

We still need to retrieve all the sequences from the accessions table generated using acc.table. Note that since we have specified download.sqs = FALSE in sq.retrieve.indirect, the sequences retrieved from GenBank are returned in a list that is stored in your global environment. If we decide to download the sequences to our working directory using download.sqs = TRUE, phruta will write all the resulting fasta files into a newly created folder 0.Sequences located in our working directory.

```
sqs.downloaded <- sq.retrieve.indirect(acc.table = acc.table, download.sqs = FALSE)
```

We are now going to make sure that we include only sequences that are reliable and from species that we are actually interested in analyzing. We are going to use the sq.curate() function for this. We will provide a list of taxonomic names to filter out incorrect sequences (filterTaxonomicCriteria argument). For instance, we could simply provide a vector of the genera that we are interested in analyzing. This vector must have a length of 1, with all the target genera being separated with | (e.g. "Callipepla|Colinus|Dendrortyx" if we were interested in only those three genera). For now, we will assume that all of the species we downloaded are relevant to the analyses (i.e. filterTaxonomicCriteria = [AZ]). Finally, since we are not downloading anything to our working directory, we need to pass our downloaded sequences (sqs.downloaded object generated above using the sq.retrieve.indirect() function) to the sqs.object argument in sq.curate().

```
sqs.curated <- sq.curate(filterTaxonomicCriteria = '[AZ]',
                         kingdom = 'animals',
                         sqs.object = sqs.downloaded,
                         removeOutliers = FALSE)
```

Running the sq.curate() function will create an object of class list including (1) the curated sequences with original names, (2) the curated sequences with species-level names (renamed_* prefix), (3) the accession numbers table (AccessionTable; @ref(tab:tw)), and (4) a summary of taxonomic information for all the species sampled in the files (@ref(tab:tw2), @ref(tab:tw3)). From here, we will simply align the sequences that we just curated. For this, we will use sq.aln() with default parameters. We're again passing the output from sq.curate(), sqs.curated, using the sqs.object argument in sq.aln().

```
sqs.aln <- sq.aln(sqs.object = sqs.curated)
```

The resulting multiple sequence alignments will be saved to the `sqs.aln` object, a list of sequence alignments. For each of the gene regions, we will have access to the original alignment (`Aln.Original`), the masked one (`Aln.Masked`), and information on the masking process. The raw and masked alignments are presented in @ref(fig:alnraw) and @ref(fig:alncur), respectively.

In total, the lines of code in this section took 6 minutes to run in my local machine (RAM: 8.5899346×10^9, CPU: Intel(R) Core(TM) i5-8257U CPU @ 1.40GHz, cores: 8, plataform: x86_64-apple-darwin17.0, R: R version 4.2.0 (2022-04-22)).

Basic phylogenetics with phruta

Phylogenetic inference with `phruta` and `RAxML`

Phylogenetic inference is conducted using the `tree.raxml()` function. To use this function, we will necessarily have to save our folders locally. We will follow the same folder structure as if we were exporting everything locally ([NEED FIGURE]). Specifically, our sequence alignments are going to be located in `2.Alignments` and we will exclusively focus on the masked ones.

```
dir.create("2.Alignments")
lapply(seq_along(sqs.aln), function(x){
  ape::write.FASTA(sqs.aln[[x]]$Aln.Masked,
                   file = paste0(
                     "2.Alignments/Masked_", names(sqs.aln)[x], ".fasta"
                     )
                   )
})
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
```

We are now ready to run RAxML with our local sequences. Note that in `tree.raxml()`, we need to indicate where the aligned sequences are located (`folder` argument), the patterns of the files in the same folder (`FilePatterns` argument; "`Masked_`" in our case), and the total of boostrap replicates. The `outgroup` argument is optional but since we are interested in calibrating our tree, we will define it (i.e. species in **Polyplectron**).

```
outgroup <- sqs.curated$Taxonomy[sqs.curated$Taxonomy$genus == 'Polyplectron',]


tree.raxml(folder = '2.Alignments',
           FilePatterns = 'Masked_',
           raxml_exec = 'raxmlHPC',
           Bootstrap = 100,
           outgroup = paste(outgroup$species_names, collapse = ",")
           )
```

```
## [1] "raxmlHPC -T 4 -f a -p 1234 -x 1234 -m GTRGAMMA -o Polyplectron_inopinatum,Polyplectron_n
```

The trees are saved in `3.Phylogeny`. Likely, the bipartitions tree, "RAxML_bipartitions.phruta", is the most relevant. `3.Phylogeny` also includes additional `RAxML`-related input and output files. Below is the resulting phylogeny from these analyses. The resulting tree is presented in @ref(fig:raxmlphylo).

Tree dating in `phruta`

Finally, let's perform tree dating in our phylogeny using secondary calibrations extracted from Scholl and Wiens (2016). I am only using this study because it has a large phylogeny but I expect to replace it in the near future. Note that `tree.dating()` requieres the user to specify where the `1.Taxonomy.csv` file is. This file is created automatically when sequences are curated using sq.curate() and results are exported to your local repository. However, since we were keeping results in the environment, we will have to export it before we can move forward.

```r
dir.create("1.CuratedSequences")
write.csv(sqs.curated$Taxonomy, '1.CuratedSequences/1.Taxonomy.csv')
```

Tree dating is performed using the `tree.dating()` function in `phruta`. We have to provide the name of the folder containing the `1.Taxonomy.csv` file created in `sq.curate()`. We also have to indicate the name of the folder containing the `RAxML_bipartitions.phruta` file. We will scale our phylogeny using `treePL`.

```r
tree.dating(taxonomyFolder = "1.CuratedSequences",
           phylogenyFolder = "3.Phylogeny",
           scale = 'treePL')
```

Running this line will result in a new folder `4.Timetree`, including the different time-calibrated phylogenies obtained (if any) and associated secondary calibrations used in the analyses. The resulting time-calibrated tree is presented in @ref(fig:timecaltree).

Advanced methods with phruta

Curating taxonomic names

You can use `taxonomy.retrieve()`, a function implemented inside `sq.curate()` in `phruta` to curate species names regardless of the kingdom. For instance, the block of code below will curate taxonomic names using the gbif taxonomic backbone. Note that the `kingdom` argument in `taxonomy.retrieve()` can be set to `NULL`, meaning that there wont be indication on the kingdom when performing taxonomic searches.

```r
phruta:::taxonomy.retrieve(species_names=c("Felis_catus", "PREDICTED:_Vulpes",
                  "Phoca_largha", "PREDICTED:_Phoca" ,
                  "PREDICTED:_Manis" , "Felis_silvestris" , "Felis_nigripes"),
                  database='gbif', kingdom=NULL)
```

```
##          kingdom   phylum    class     order   family genus          species
## 1       Animalia Chordata Mammalia Carnivora  Felidae Felis      Felis catus
## 2 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 3       Animalia Chordata Mammalia Carnivora Phocidae Phoca     Phoca largha
## 4 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 5 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 6       Animalia Chordata Mammalia Carnivora  Felidae Felis Felis silvestris
## 7       Animalia Chordata Mammalia Carnivora  Felidae Felis   Felis nigripes
```

However, `gbif` is efficient for retrieving accurate taxonomy when we provide details on the `kingdom`. Given that all the species we're interested in are animals, we could just use the following block of code to curate taxonomic names.

```
phruta:::taxonomy.retrieve(species_names=c("Felis_catus", "PREDICTED:_Vulpes",
                  "Phoca_largha", "PREDICTED:_Phoca" ,
                  "PREDICTED:_Manis" , "Felis_silvestris" , "Felis_nigripes"),
                  database='gbif', kingdom='animals')
```

```
##          kingdom   phylum    class     order   family genus          species
## 1       Animalia Chordata Mammalia Carnivora  Felidae Felis      Felis catus
## 2 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 3       Animalia Chordata Mammalia Carnivora Phocidae Phoca     Phoca largha
## 4 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 5 incertae sedis     <NA>     <NA>      <NA>     <NA>  <NA>             <NA>
## 6       Animalia Chordata Mammalia Carnivora  Felidae Felis Felis silvestris
## 7       Animalia Chordata Mammalia Carnivora  Felidae Felis   Felis nigripes
```

Depending on your sampling, you could also do the same for plants by using `plants` in thr `kingdom` argument instead of `animals`. Now, what if we were interested in following other databases to retrieve taxonomic information for the species in our database? The latest version of `phruta` allow users to select the desired database. The databases follow the `taxize::classification()` function. Options are: `ncbi`, `itis`, `eol`, `tropicos`, `nbn`, `worms`, `natserv`, `bold`, `wiki`, and `pow`. Please select only one. Note that the `gbif` option in `taxize::classification()` is replaced by the internal `gbif` in `phruta`.

Now, let's assume that we were interested in curating our database using `itis`:

```
phruta:::taxonomy.retrieve(species_names=c("Felis_catus", "PREDICTED:_Vulpes",
                    "Phoca_largha", "PREDICTED:_Phoca" ,
                    "PREDICTED:_Manis" , "Felis_silvestris" , "Felis_nigripes"),
                    database='itis')
```

Creating taxonomic constraints in ***`phruta`***

Users sometimes need to generate tree constrains. `phruta` can automatically generate trees in accordance to taxonomy and a backbone topology. More complex tree constraints can be generated other software including `TACT: Taxonomic Addition for Complete Trees`. For `phruta`, we divide constraint trees into two classes: (1) ingroup+outgroup and (2) particular clades.

ingroup + outgroup constrains [This section needs more work]

In this constraint type, `phruta` will create monophyletic groups using `tree.constraint()` for each of the taxonomic groups in the database (for selected target columns). Users will need to provide a path to the `1.Taxonomy.csv` file created using the `sq.curate()` function. Finally, `tree.constraint()` will generate tree with the same topology provided in the `Topology` argument. The user will provide the species names of the outgroup taxa as a vector of string that should fully match the names in the taxonomy file.

```
tree.constraint(
                 taxonomy_folder = "1.CuratedSequences",
                 targetColumns = c("kingdom", "phylum", "class", "order",
                                   "family", "genus", "species_names"),
                 Topology = "((ingroup), outgroup);",
                 outgroup = outgroup$species_names[2]
 )
```

Constrains on particular clades In this constraint type, `tree.constraint()` will create a constraint tree for particular clades. Note that the key aspect here is the `Topology` argument. It is a newick tree. For instance, let's assume that we only need to create a tree constraining the monophyly within two genera and their sister relationships:

```
tree.constraint( taxonomy_folder = "1.CuratedSequences",
                 targetColumns = c("kingdom", "phylum", "class",
                                   "order", "family", "genus", "species_names"),
                 Topology = "((Callipepla), (Polyplectron));"
 )
```

Running PartitionFinder in `phruta`

With the current version of `phruta`, users are able to run `PartitionFinder` v1 within `R`. For this, users should provide the name of the folder where the alignments are stored, a particular pattern in the file names (`Masked_` in our case), and which models will be run in `PartitionFinder`. This function will download `PartitionFinder`, generate the input files, and run it all within R. The output files will be in a new folder within the working directory.

```
sq.partitionfinderv1(folderAlignments = "2.Alignments",
                     FilePatterns = "Masked_",
                     models = "all"
 )
```

Unfortunately, the output files are not integrated with the current `phruta` pipeline. This will be part of a new release. However, users can still perform gene-based partitioned analyses within `RAxML` or can use PartitionFinder's output files to inform their own analyses outside `phruta`.

Partitioned analyses in RAxML

Users can now run partitioned analyses in `RAxML` within `phruta`. This approach is implemented by setting the `partitioned` argument in `tree.raxml` to `TRUE`. For now, partitions are based on the genes are being analyzed. The same model is used to analyze each partition. More details on partitioned analyses can be customized by passing arguments in `ips::raxml`.

```
tree.raxml(folder = "2.Alignments", FilePatterns = "Masked_",
           raxml_exec = "raxmlHPC", Bootstrap = 100,
           outgroup = paste(outgroup$species_names, collapse = ","),
           partitioned = TRUE
)
```

```
## [1] "raxmlHPC -T 4 -f a -p 1234 -x 1234 -m GTRGAMMA -o Polyplectron_inopinatum,Polyplectron_n
```

Identifying rogue taxa

`phruta` can help users run `RogueNaRok` implemented in the `Rogue` R package. Users can then examine whether rogue taxa should be excluded from the analyses. `tree.roguetaxa()` uses the bootstrap trees generated using the `tree.raxml()` function along with the associated best tree to identify rogue taxa.

```
tree.roguetaxa(folder = "3.Phylogeny")
```

Reproducibility with phruta Users can choose to share the script they used to run the analyses (e.g. assemble their molecular dataset) and the associated workspace.

Performance

http://jboyd.net/Taxo/Odontophoridae.pdf

Hosner, P.A., E.L. Braun, and R.T. Kimball (2015a), Land connectivity changes and global cooling shaped the colonization history and diversification of New World quail (Aves: Galliformes: Odontophoridae), J. Biogeogr. 42, 1883-1895.

Crowe, T.M., R.C.K. Bowie, P. Bloomer, T.G. Mandiwana, T.A.J. Hedderson, E. Randi, S. Pereira, and J. Wakeling (2006a), Phylogenetics, biogeography and classification of, and character evolution in, gamebirds (Aves: Galliformes): Effects of character exclusion, data partitioning and missing data, Cladistics 22, 495-532.

Crowe, T.M., P. Bloomer, E. Randi, V. Lucchini, R. Kimball, E. Braun, and J.G. Groth (2006b), Supra-generic cladistics of landfowl (Order Galliformes), Acta Zool. Sinica 52, S358-S361.

Cohen, C., J.L. Wakeling, T.G. Mandiwana-Neudani, E. Sande, C. Dranzoa, T.M. Crowe, and R.C.K. Bowie (2012), Phylogenetic affinities of evolutionarily enigmatic African galliforms: the Stone Partridge Ptilopachus petrosus and Nahan's Francolin Francolinus nahani, and support for their sister relationship with New World quails, Ibis 154, 768-780.

Table 1: Top six genes sampled in GenBank for species in Odontophoridae and Polyplectron.

Gene	Sampled in N species	PercentOfSampledSpecies
NADH dehydrogenase subunit 2	90	98.90110
12S ribosomal RNA	27	29.67033
eukaryotic elongation factor 2	25	27.47253
NADH dehydrogenase subunit 5	18	19.78022
cytochrome b	16	17.58242
cytochrome oxidase subunit 1	10	10.98901

Figure 1: Quail placeholder. Phyto by Brent Myers

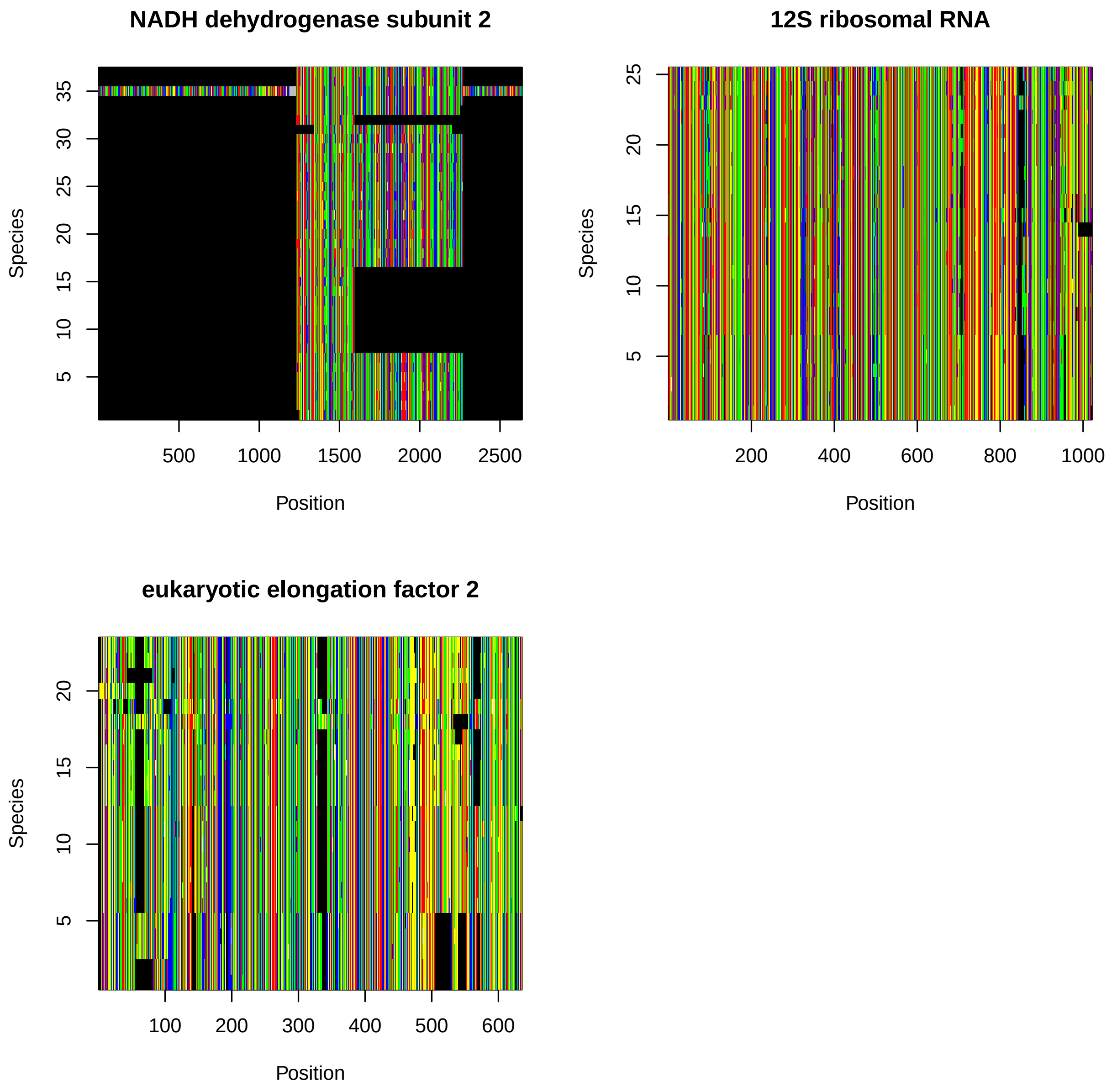

Figure 2: Raw alignments for gene regions sampled in more than 20% of the species in GenBank

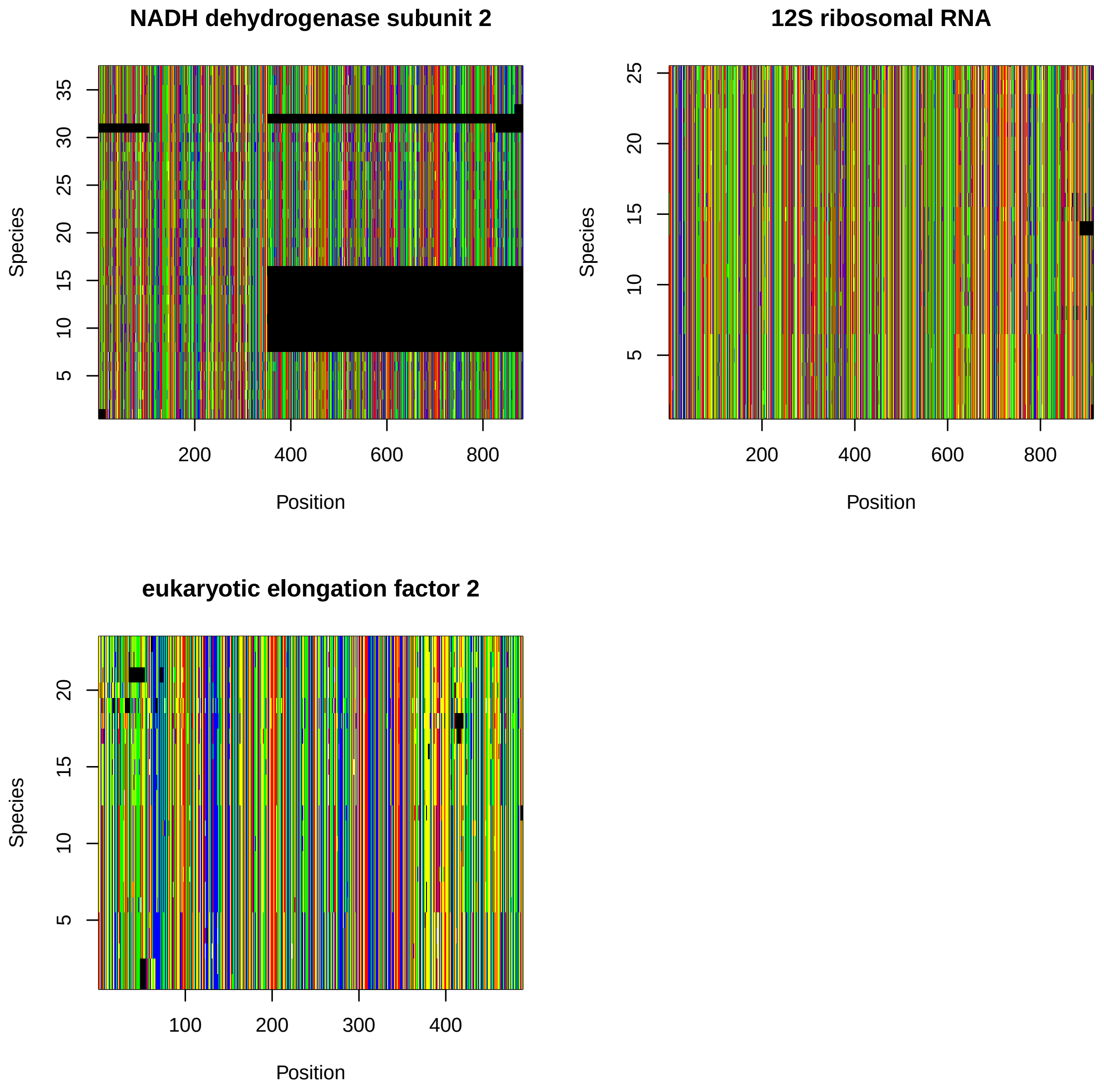

Figure 3: Curated alignments for gene regions sampled in more than 20% of the species in GenBank

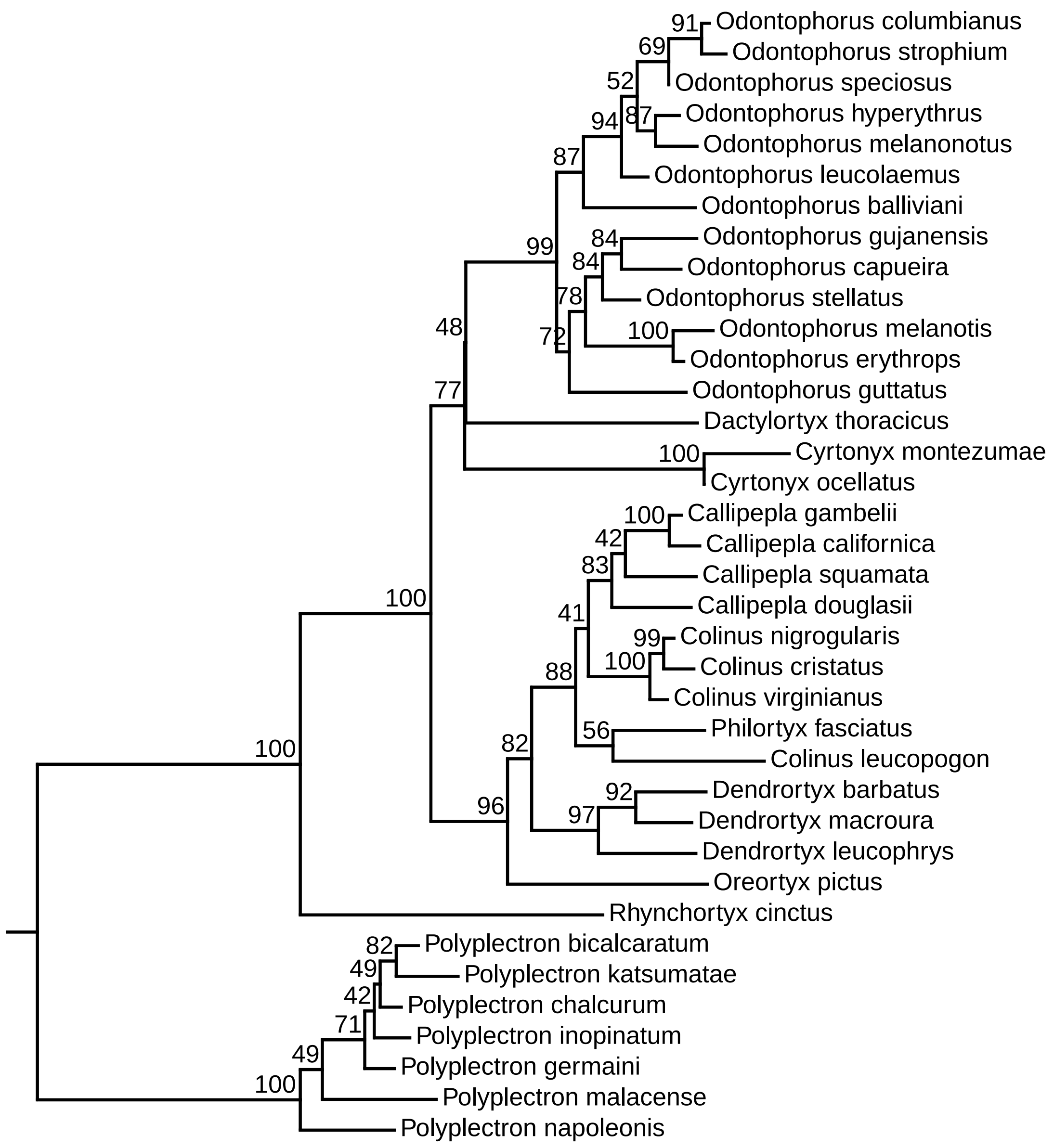

Figure 4: RAxML phylo

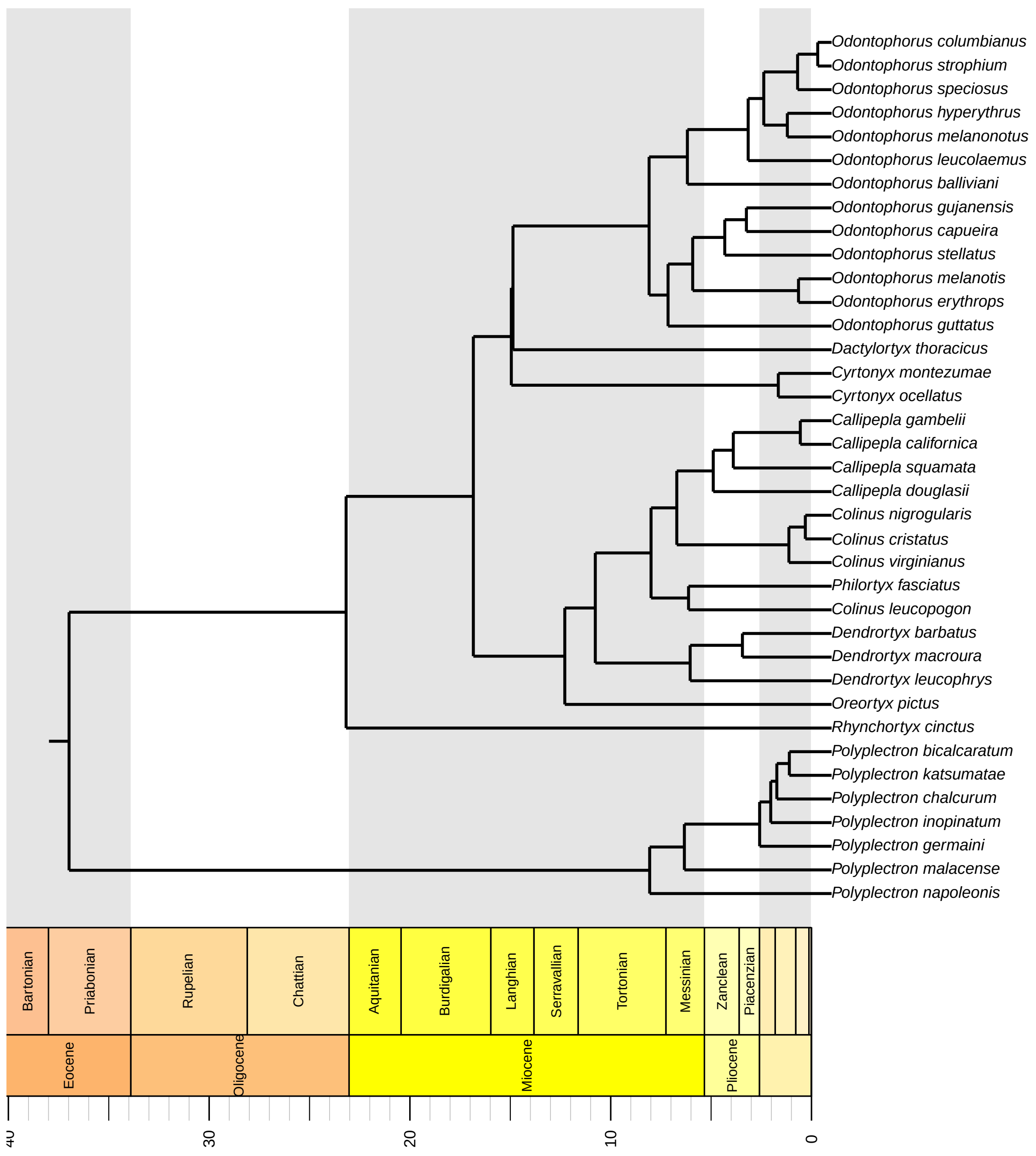

Figure 5: Time-calibrated phylogeny phylo

Table 2: Summary of potential accession numbers for the species in Odontophoridae, our ingroup, and Polyplectron, outgroup genus. This list of sequences has not been curated yet.

Species	Acc	gene
Callipepla californica	MZ476322	NADH dehydrogenase subunit 2
Callipepla gambelii	MZ476314	NADH dehydrogenase subunit 2
Colinus virginianus	EU166949	NADH dehydrogenase subunit 2
Colinus cristatus	AF222544	NADH dehydrogenase subunit 2
Colinus nigrogularis	KR732857	NADH dehydrogenase subunit 2
Dendrortyx barbatus	KR732856	NADH dehydrogenase subunit 2
Philortyx fasciatus	KR732855	NADH dehydrogenase subunit 2
Rhynchortyx cinctus	KR732850	NADH dehydrogenase subunit 2
Cyrtonyx montezumae	KR732849	NADH dehydrogenase subunit 2
Oreortyx pictus	KR732848	NADH dehydrogenase subunit 2
Odontophorus leucolaemus	KR732847	NADH dehydrogenase subunit 2
Odontophorus speciosus	KR732846	NADH dehydrogenase subunit 2
Odontophorus erythrops	KR732845	NADH dehydrogenase subunit 2
Odontophorus guttatus	KR732844	NADH dehydrogenase subunit 2
Odontophorus gujanensis	KR732843	NADH dehydrogenase subunit 2
Odontophorus capueira	KR732842	NADH dehydrogenase subunit 2
Odontophorus stellatus	KR732841	NADH dehydrogenase subunit 2
Dactylortyx thoracicus	KR732840	NADH dehydrogenase subunit 2
Dendrortyx macroura	KR732839	NADH dehydrogenase subunit 2
Callipepla squamata	KR732838	NADH dehydrogenase subunit 2
Callipepla douglasii	KR732837	NADH dehydrogenase subunit 2
Colinus leucopogon	KC556543	NADH dehydrogenase subunit 2
Dendrortyx leucophrys	KC556066	NADH dehydrogenase subunit 2
Cyrtonyx ocellatus	KC556060	NADH dehydrogenase subunit 2
Odontophorus balliviani	KC556524	NADH dehydrogenase subunit 2
Odontophorus columbianus	KC556517	NADH dehydrogenase subunit 2
Odontophorus strophium	KC556515	NADH dehydrogenase subunit 2
Odontophorus melanonotus	KC556513	NADH dehydrogenase subunit 2
Odontophorus hyperythrus	KC556512	NADH dehydrogenase subunit 2
Odontophorus melanotis	KC556507	NADH dehydrogenase subunit 2
Polyplectron inopinatum	EF569482	NADH dehydrogenase subunit 2
Polyplectron napoleonis	EF569481	NADH dehydrogenase subunit 2
Polyplectron chalcurum	EF569480	NADH dehydrogenase subunit 2
Polyplectron bicalcaratum	EF569479	NADH dehydrogenase subunit 2
Polyplectron malacense	DQ768268	NADH dehydrogenase subunit 2
Polyplectron germaini	DQ768266	NADH dehydrogenase subunit 2
Polyplectron katsumatae	KC778823	NADH dehydrogenase subunit 2
Cyrtonyx montezumae	KR732830	12S ribosomal RNA
Dactylortyx thoracicus	KR732829	12S ribosomal RNA
Oreortyx pictus	KR732828	12S ribosomal RNA
Odontophorus erythrops	KR732827	12S ribosomal RNA
Odontophorus gujanensis	KR732826	12S ribosomal RNA
Odontophorus stellatus	KR732825	12S ribosomal RNA
Odontophorus capueira	KR732824	12S ribosomal RNA
Odontophorus speciosus	KR732823	12S ribosomal RNA
Odontophorus leucolaemus	KR732822	12S ribosomal RNA
Odontophorus balliviani	KR732821	12S ribosomal RNA
Dendrortyx macroura	KR732820	12S ribosomal RNA

Table 3: Accession numbers for the retrieved sequences. This dataset has been curated.

OriginalNames	AccN	Species	file
MZ476322 Callipepla californica	MZ476322	Callipepla_californica	NADH dehydrogenase s
MZ476314 Callipepla gambelii	MZ476314	Callipepla_gambelii	NADH dehydrogenase s
EU166949 Colinus virginianus	EU166949	Colinus_virginianus	NADH dehydrogenase s
AF222544 Colinus cristatus	AF222544	Colinus_cristatus	NADH dehydrogenase s
KR732857 Colinus nigrogularis	KR732857	Colinus_nigrogularis	NADH dehydrogenase s
KR732856 Dendrortyx barbatus	KR732856	Dendrortyx_barbatus	NADH dehydrogenase s
KR732855 Philortyx fasciatus	KR732855	Philortyx_fasciatus	NADH dehydrogenase s
KR732850 Rhynchortyx cinctus	KR732850	Rhynchortyx_cinctus	NADH dehydrogenase s
KR732849 Cyrtonyx montezumae	KR732849	Cyrtonyx_montezumae	NADH dehydrogenase s
KR732848 Oreortyx pictus	KR732848	Oreortyx_pictus	NADH dehydrogenase s
KR732847 Odontophorus leucolaemus	KR732847	Odontophorus_leucolaemus	NADH dehydrogenase s
KR732846 Odontophorus speciosus	KR732846	Odontophorus_speciosus	NADH dehydrogenase s
KR732845 Odontophorus erythrops	KR732845	Odontophorus_erythrops	NADH dehydrogenase s
KR732844 Odontophorus guttatus	KR732844	Odontophorus_guttatus	NADH dehydrogenase s
KR732843 Odontophorus gujanensis	KR732843	Odontophorus_gujanensis	NADH dehydrogenase s
KR732842 Odontophorus capueira	KR732842	Odontophorus_capueira	NADH dehydrogenase s
KR732841 Odontophorus stellatus	KR732841	Odontophorus_stellatus	NADH dehydrogenase s
KR732840 Dactylortyx thoracicus	KR732840	Dactylortyx_thoracicus	NADH dehydrogenase s
KR732839 Dendrortyx macroura	KR732839	Dendrortyx_macroura	NADH dehydrogenase s
KR732838 Callipepla squamata	KR732838	Callipepla_squamata	NADH dehydrogenase s
KR732837 Callipepla douglasii	KR732837	Callipepla_douglasii	NADH dehydrogenase s
KC556543 Colinus leucopogon	KC556543	Colinus_leucopogon	NADH dehydrogenase s
KC556066 Dendrortyx leucophrys	KC556066	Dendrortyx_leucophrys	NADH dehydrogenase s
KC556060 Cyrtonyx ocellatus	KC556060	Cyrtonyx_ocellatus	NADH dehydrogenase s
KC556524 Odontophorus balliviani	KC556524	Odontophorus_balliviani	NADH dehydrogenase s
KC556517 Odontophorus columbianus	KC556517	Odontophorus_columbianus	NADH dehydrogenase s
KC556515 Odontophorus strophium	KC556515	Odontophorus_strophium	NADH dehydrogenase s
KC556513 Odontophorus melanonotus	KC556513	Odontophorus_melanonotus	NADH dehydrogenase s
KC556512 Odontophorus hyperythrus	KC556512	Odontophorus_hyperythrus	NADH dehydrogenase s
KC556507 Odontophorus melanotis	KC556507	Odontophorus_melanotis	NADH dehydrogenase s
EF569482 Polyplectron inopinatum	EF569482	Polyplectron_inopinatum	NADH dehydrogenase s
EF569481 Polyplectron napoleonis	EF569481	Polyplectron_napoleonis	NADH dehydrogenase s
EF569480 Polyplectron chalcurum	EF569480	Polyplectron_chalcurum	NADH dehydrogenase s
EF569479 Polyplectron bicalcaratum	EF569479	Polyplectron_bicalcaratum	NADH dehydrogenase s
DQ768268 Polyplectron malacense	DQ768268	Polyplectron_malacense	NADH dehydrogenase s
DQ768266 Polyplectron germaini	DQ768266	Polyplectron_germaini	NADH dehydrogenase s
KC778823 Polyplectron katsumatae	KC778823	Polyplectron_katsumatae	NADH dehydrogenase s
KR732830 Cyrtonyx montezumae	KR732830	Cyrtonyx_montezumae	12S ribosomal RNA
KR732829 Dactylortyx thoracicus	KR732829	Dactylortyx_thoracicus	12S ribosomal RNA
KR732828 Oreortyx pictus	KR732828	Oreortyx_pictus	12S ribosomal RNA
KR732827 Odontophorus erythrops	KR732827	Odontophorus_erythrops	12S ribosomal RNA
KR732826 Odontophorus gujanensis	KR732826	Odontophorus_gujanensis	12S ribosomal RNA
KR732825 Odontophorus stellatus	KR732825	Odontophorus_stellatus	12S ribosomal RNA
KR732824 Odontophorus capueira	KR732824	Odontophorus_capueira	12S ribosomal RNA
KR732823 Odontophorus speciosus	KR732823	Odontophorus_speciosus	12S ribosomal RNA
KR732822 Odontophorus leucolaemus	KR732822	Odontophorus_leucolaemus	12S ribosomal RNA
KR732821 Odontophorus balliviani	KR732821	Odontophorus_balliviani	12S ribosomal RNA
KR732820 Dendrortyx macroura	KR732820	Dendrortyx_macroura	12S ribosomal RNA
KR732819 Philortyx fasciatus	KR732819	Philortyx_fasciatus	12S ribosomal RNA

Table 4: Taxonomic information for the retrieved species

kingdom	phylum	class	order	family	genus	species
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla	Callipepla californica
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla	Callipepla gambelii
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus	Colinus virginianus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus	Colinus cristatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus	Colinus nigrogularis
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx	Dendrortyx barbatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Philortyx	Philortyx fasciatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Rhynchortyx	Rhynchortyx cinctus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Cyrtonyx	Cyrtonyx montezumae
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Oreortyx	Oreortyx pictus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus leucolaemus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus speciosus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus erythrops
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus guttatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus gujanensis
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus capueira
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus stellatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dactylortyx	Dactylortyx thoracicus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx	Dendrortyx macroura
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla	Callipepla squamata
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla	Callipepla douglasii
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus	Colinus leucopogon
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx	Dendrortyx leucophrys
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Cyrtonyx	Cyrtonyx ocellatus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus balliviani
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus columbianus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus strophium
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus melanonotus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus hyperythrus
Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus	Odontophorus melanotis
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron inopinatum
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron napoleonis
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron chalcurum
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron bicalcaratum
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron malacense
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron germaini
Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron	Polyplectron katsumatae

Table 5: Taxonomic sampling across gene regions

species_names	kingdom	phylum	class	order	family	genus
Callipepla_californica	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla
Callipepla_douglasii	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla
Callipepla_gambelii	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla
Callipepla_squamata	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Callipepla
Colinus_cristatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus
Colinus_leucopogon	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus
Colinus_nigrogularis	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus
Colinus_virginianus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Colinus
Cyrtonyx_montezumae	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Cyrtonyx
Cyrtonyx_ocellatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Cyrtonyx
Dactylortyx_thoracicus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dactylortyx
Dendrortyx_barbatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx
Dendrortyx_leucophrys	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx
Dendrortyx_macroura	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Dendrortyx
Odontophorus_balliviani	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_capueira	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_columbianus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_erythrops	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_gujanensis	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_guttatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_hyperythrus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_leucolaemus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_melanonotus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_melanotis	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_speciosus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_stellatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Odontophorus_strophium	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Odontophorus
Oreortyx_pictus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Oreortyx
Philortyx_fasciatus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Philortyx
Polyplectron_bicalcaratum	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_chalcurum	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_germaini	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_inopinatum	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_katsumatae	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_malacense	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Polyplectron_napoleonis	Animalia	Chordata	Aves	Galliformes	Phasianidae	Polyplectron
Rhynchortyx_cinctus	Animalia	Chordata	Aves	Galliformes	Odontophoridae	Rhynchortyx

Table 6: Results of RogueNaRock

num	taxNum	taxon	rawImprovement	IC
0	NA	NA	NA	355.5942
1	4	Colinus_leucopogon	16.74831	372.3425