The University of South Bohemia in České Budějovice

Faculty of Science

A semi-automatic identification pipeline for species identification and phylogenetic comparative analyses: tropical butterflies as a case example

Bioinformatics project

Tran Doan Chau

Supervisor: RNDr. Pável Matos-Maraví, Ph.D., Biology Centre, CAS

České Budějovice 2023

Contents

Abstract

DNA barcoding is a method that uses a short, standardized region of DNA as a diagnostic marker to identify species. Due to fast evolving substitution rates, every species has its unique barcode sequence, which can be compared to a reference library of such sequences. The barcode identification approach consists of comparing the query sequences against the subject sequences, usually by estimating the percentage identities via BLAST, or by phylogenetic relationships and monophyly, via quick neighbor joining phylogeny estimations. Users can easily identify sequences via online systems such as BOLD. However, there are still some limitations, for example, the identification of species depends largely on the amount of information available on the databases, which is typically biased towards the most studied organisms such as charismatic groups or the ones "important" for humans such as pathogens. Further, due to computational resources, the online identification engines have restrictions on the number of sequences to be analyzed in batch, e.g., a maximum of 50 sequences at a time in BOLD. This project aims to create a semi-automatic identification pipeline using the command line to be able to process thousands of sequences (queries) in a single run, with the overall aim to create user-friendly output files depicting BLAST percentage identities and robust phylogenetic relationships information using a maximum likelihood approach. Our work provides efficient pipelines that can be possibly further expanded into a more automated program where end users state in the beginning of the approach what the level of percent identity is required to cluster species (95% or more) and the possibility to generate backups and updates in real-time with public databases and own sequences that were already identified, which can then be added safely to the reference database for future reference.

1 Introduction

DNA barcoding is a method that uses a short, standardized region of DNA as a diagnostic marker to identify species. Due to fast evolving substitution rates, every species has its unique barcode sequence, which can be compared to a reference library of such sequences. Markers used for DNA barcoding are called barcode loci and the choice of these depends on the group of study organisms. Different regions of the genome can be used as barcodes for different taxonomic groups, and for animals, the mitochondrial cytochrome c oxidase subunit I (COI) locus is the most widely used barcode (Hebert et al. 2003).

Barcodes are usually short DNA sequences, around 400–800 bp, intended for straightforward generation and characterization across all species globally. An extensive online digital repository of these barcodes is envisioned to function as a benchmark, allowing the DNA barcode sequence of an unidentified sample from the forest, garden, or market to be compared and matched (Kress and Erickson 2008). Examples of such digital repositories include the National Center for Biotechnology Information (NCBI) database available at https://www.ncbi.nlm.nih.gov/ or the Barcode of Life Data Systems (BOLD) system, accessible at http://www.boldsystems.org. The barcode identification approach consists of comparing the query sequences against the subject sequences, usually by estimating the percentage identities via BLAST, or by phylogenetic relationships and monophyly, via quick neighbor joining phylogeny estimations.

Although online systems such as BOLD are user-friendly, there are still some limitations. For example, the identification of species depends largely on the amount of information available on the databases, which is typically biased towards charismatic organisms that are easy to sample. Further, due to computational resources, the online identification engines have restrictions on the number of sequences to be analyzed in batch, e.g., a maximum of 50 sequences at a time in BOLD. Finally, the neighbor joining phylogenetic approach while amenable with limited computational resources, may suffer from methodological biases generating incorrect phylogenetic inferences.

The motivation of this project is to create a semi-automatic identification pipeline using the command line to be able to process thousands of sequences (queries) in a single run, with the overall aim to create user-friendly output files depicting BLAST percentage identities and robust phylogenetic relationships information using a maximum likelihood approach. With the approach, we asked what the species diversity is in this region as a preliminary step for future projects to understand the distribution of correlation of species traits with the environment across habitats. We expect that a significant proportion of species sampled in the region is not represented in public databases because we focus on butterfly groups with little taxonomic understanding (e.g., Hesperiidae).

2 Work aim

The goal of this project is to identify the species names of query barcode sequences using the command line interface to ensure a semi-automatic approach complementary to existing online identification engines (e.g., BOLD). The approach consists of retrieving large amounts of sequences of the study groups at a large taxonomic (e.g., family) and restricted geographical scopes (e.g., biogeographical region). Then, building a local reference database using the Blast package to estimate sequence similarities between query and reference sequences. Afterward, Python scripts are used to search for the likely species names per query sequence, which are exported as two user-friendly data frames: one with only best similarity scores ≥ 95% p-identity; one with the remaining BLAST searches having similarity scores less than 95%. The query sequences are then aligned, and a phylogenetic tree is inferred displaying all the study sequences. Finally, for those query sequences that were not reliably identified (i.e., < 95% similarity), a Python script finds a list of closely related sequences within the phylogeny tree to determine the likely sister species.

3 Materials and Methods

3.1 Study sites and species

Samples were collected in the premontane tropical rainforest (~400 –800 m) near a national park named ‘Área de Conservación Regional Cordillera Escalera’ located in Tarapoto, Peru. The research team including Pável Matos, Daniel Linke, and local collaborators, visited the locality twice, once during the rainy season from October 2021 to February 2022, and twice during the dry season of June to September 2022. Local conditions ranged from moist and shady valleys, semi-open permaculture plantations, closed secondary forest cut by walking paths to dry, windy hilltops with xerophilic plants. The butterflies were caught when encountered during field walking using entomological nets. Butterfly sampling was random and did not represent true species diversity and composition at the study location.

3.2 DNA sequencing

The total number of specimens that were analyzed was 1,555. Total DNA was extracted from two butterfly legs per specimen using the QIAGEN’s DNeasy kit by a technician. Amplification of the mitochondrial cytochrome c oxidase subunit I (COI) gene was performed using published primers and PCR protocols (Matos-Maraví et al. 2013). DNA sequencing was conducted by the company Macrogen Europe BV (Amsterdam, The Netherlands). The resulting chromatograms and DNA sequences were inspected and edited accordingly using the program Geneious Prime 2023.2.1 (http://www.geneious.com/).

3.3 Molecular species identification

3.3.1 Command line interface to retrieve BOLD databases

A command line interface was used to retrieve the study sequences from the families Hesperiidae, Nymphalidae, Papilioneidae, Pieridae, and Riodinidae from the Barcode of Life Data Systems (BOLD) system, accessible at http://www.boldsystems.org. This bioinformatic pipeline is called “BOLD-CLI”, as detailed by Nugent (2019) (Appendix code 1). Subsequently, multiple bash command lines were executed to curate data procedures, such as renaming all sequence headers while storing original names for later retrieval and eliminating sequences that do not belong to the barcode region “COI-5P”. Furthermore, local BLAST databases were built in the Metacentrum environment (the Czech National Grid Organization, https://metavo.metacentrum.cz/) using the retrieved COI sequences from BOLD (Appendix code 2). Lastly, BLAST searches were performed using the command ‘*blastn’* (Altschul et al. 1990) with our samples as queries against the reference database (Appendix code 3). After BLAST, a Bash command line was used to convert the output into an Excel file (Appendix table 1), wherein the ‘sseqid’ was subsequently replaced by the original names previously stored from “BOLD-CLI” for further processing.

1	qseqid	query or source (gene) sequence id
2	sseqid	subject or target (reference genome) sequence id
3	pident	percentage of identical positions

4	length	alignment length (sequence overlap)
5	mismatch	number of mismatches
6	gapopen	number of gap openings

Table 1. BLASTn tabular output format 6 showed the parameters of the command line '*blastn'*

3.3.2 Phylogenetic tree

To infer and visualize the evolutionary relationships among the studied sequences, we inferred a maximum likelihood phylogenetic tree. The COI sequences were aligned using the Multiple Alignment using Fast Fourier Transform (MAFFT) tool v7.520 (Katoh and Standley 2013), which matched homologous positions along the COI gene. The phylogenetic tree was constructed using the aligned dataset in the IQ-TREE multicore software version 2.2.0 (Minh et al. 2020). This allowed the system to explore different potential tree topologies and choose the best based on our data using the Ultrafast Bootstrap Approximation (Hoang et al., 2018) with 1,000 replicates for statistical support values. To ensure the accuracy of model, the COI alignment was partitioned into codon positions, allowing the program to find the best scheme and substitution models using the commands `-m MFP –merge` via ModelFinder (Appendix code 4 and Appendix code 5).

3.4 Finding best matches

All Python scripts imported the package "pandas" to read the data frames for data manipulation and analysis.

A customized Python program was applied to filter the best matches out of each query sequence. This was done aiming to target any potential misidentifications in the database (reference error), as for example, a taxonomically difficult group may be misidentified by taxonomists. First, the program calculates the total occurrences of best-matched sequences (i.e., > 95% p-identity) by query sequence. Second, the program calculates the frequency of the species names attributed to best-matched reference sequences and report the species name with the highest frequency (i.e., the ratio of the number of matches with a species name divided by the total number of best matches with p-identity > 95%). Finally, the best match together with its frequency in the database is reported as the most likely species name for each query sequence (Appendix code 6).

After our stringent filter of p-identities >95%, some query sequences could not be reliably matched with a species name available in the reference database. To provide to the end-user a solid closest match for those query sequences, a customized Python program was created to find missing sequences in the output to identify data loss (Appendix code 7). Subsequently, another Biopython class called "FindingCloselyRelated" was programmed to read the inferred maximum-likelihood phylogenetic tree by '*Phylo'* module (Cock et al. 2009) and find the closest related sequences. The program searches for the minimum distances and returns the list of sequences that share a most recent common ancestor to each query sequence with p-identities <95%. The scripts were written using the libraries "*tqdm*" and "*time*" to measure the time and progression of the processes; "torch" to run the script with GPU to reduce the time consumption for the processes. This script was customized to iterate the list of all sequence with p-identities

<95% and executed the function ‘get_closest_taxa’ for each query sequence. The result was extracted into a dictionary, and further into a data frame with the query sequence and its closely related sequences within the phylogenetic tree (Appendix code 8). Furthermore, a list of species names of all closely related sequences is reported in the data frame (Appendix code 9). In this way, the end-user can safely identify all their query sequences at least to the genus level.

4 Results

There were 1,555 query sequences included in the study, of which 1 sequence was not a butterfly sequence but a *Wolbachia* insect endosymbiont, 1,383 sequences had solid identifications at the species level (217 butterfly species) (Appendix table 2), whereas 171 query sequences were identified at least at the genus level and may represent 78 additional species.

Tables	Total_queries
Subset_blast_table	1,554
Blast_95_table	1,383
Blast_table	~171

Table 2. Total query sequences analyzed with our semi-automatic pipeline: Subset_blast_table (all best matching butterfly query sequences); Blast_95_table (all best matching query sequences with p-ident ≥ 95%); Blast_table (all best matching query sequences with p-ident < 95%)

4.1 Phylogenetic relationships

To compare the evolutionary relationships among the sampled sequences, we inferred a maximum likelihood phylogenetic tree using the COI sequences. ModelFinder estimated that the first and second coding positions should be merged, and the third coding position should be analyzed in a different partition. Both partitions had the “*TIM2*” as the best-fit substitution model.

4.2 Taxonomic identifications of query sequences

The first Python program subset the best matches per query sequences from the output from BLAST. The result was that there were 1,383 sequences identified with high p-ident values (≥ 95%) and 171 sequences with p-ident values lower than 95%. These two outputs are reported respectively in the Table 3 and Table 4. Out of 267 identified unique species within 1,554 butterfly query sequences, there were 266 unique species identified with high similarity rate (≥ 95%) for 1,383 query sequences (Appendix table 3) and ~77 unique species identified with low similarity rate (<95%) for 171 query sequences.

species	qseqid	pident	samples	frequency
Phocides_Burns01	EC00031420-DL_857	95.082	4	100
Telegonus_fulminator	EC00031420-DL_860	95.11	1	100

Cogia_stylites	EC00022554-PM_533	95.129	1	100
Cogia_calchas	A259a_T7promoter.ab1	95.139	1	100
Cogia_stylites	EC00022554-PM_503	95.142	1	100
Cogia_stylites	EC00022554-PM_506	95.142	1	100
Cogia_stylites	EC00022558-PM_594	95.146	1	100
Cecropterus_jalapus	EC00031423-DL_995	95.151	7	100
Aguna_claxon	EC00031418-DL_552	95.156	6	100
Celaenorrhinus_eligius	EC00031418-DL_590	95.207	1	100

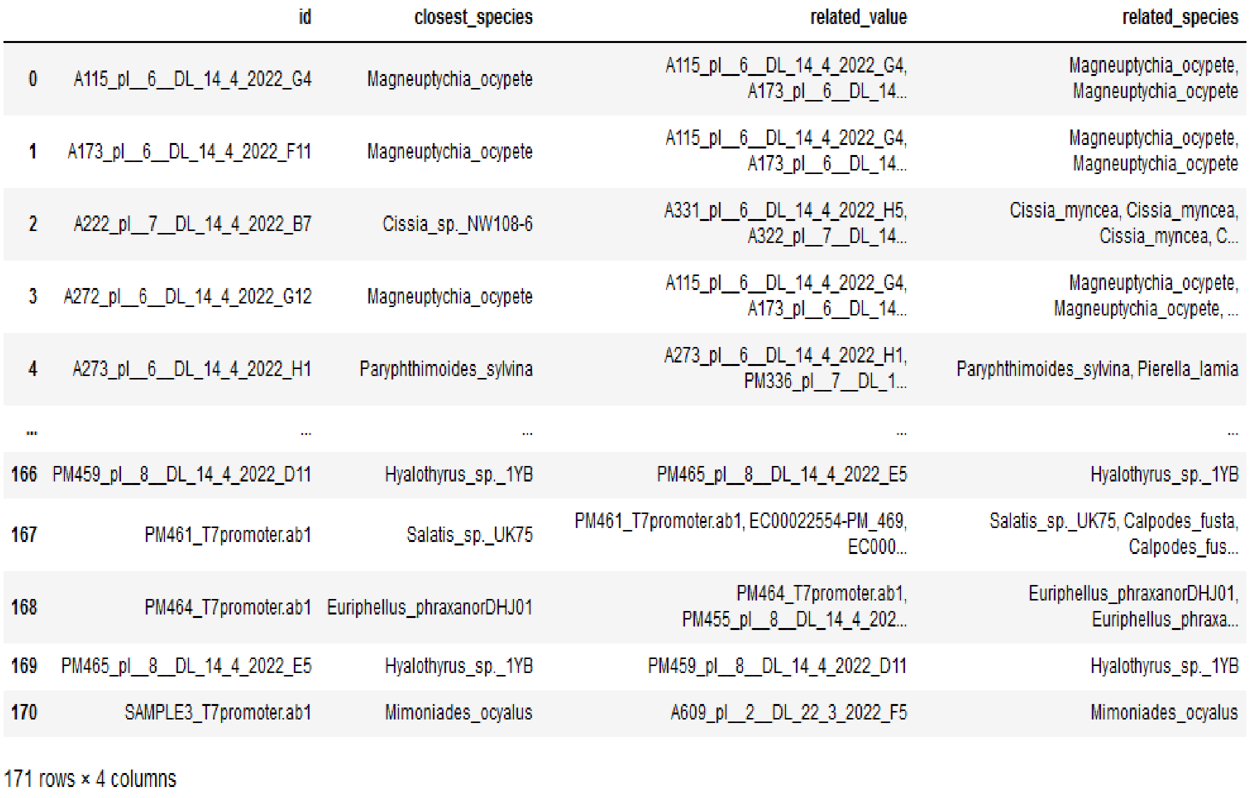
Table 3. The table displayed the partial result of finding the best matching subsequences of pident ≥ 95% with the highest p-ident and frequency.

species	qseqid	pident	samples	frequency
Oeneis_ammon	EC00022554-PM_508	85.993	2	100
Toxidia_rietmanni	EC00031418-DL_491	88.148	1	100
Toxidia_andersoni	EC00023625-DL_DL0356	88.462	1	100
Pseudargynnis_hegemone	EC00023628-DL_DL0140	88.789	1	100
Spicauda_simplicius	EC00031423-DL_978	88.929	1	100
Ectomis_orpheus	EC00031425-DL_1159	89.189	1	100
Autochton_longipennis	EC00031423-DL_977	89.931	1	100
Cercyonis_oetus	EC00031418-DL_520	90.554	2	100
Mylothris_schumanni_uniformis	EC00022558-PM_600	90.651	1	100
Autochton_longipennis	EC00031425-DL_1178	90.664	1	100

Table 4. The table showed the partial result of best matching subsequences of less than 95% pident with the highest p-ident and frequency.

The second script was applied to extract a list of all 1,555 query sequences names and compare them with the output from the raw table in the first Python program to corroborate that our approach worked as expected. The result indicated that one query sequence was missing: the specimen ID “PM443_pl__8__DL_14_4_2022_C7”. This sequence had indeed a very long branch in the maximum likelihood phylogeny tree, and subsequence BLAST searches against the entire NCBI database suggested that its sequence belonged to the butterfly endosymbiont *Wolbachia*. This result is not surprising, as this bacterium typically infects butterflies in the wild. This query sequence was then removed from the datasets, and we re-ran our scripts.

The last scripts iterated each sequence id in the datasets where p-ident was less than 95% and identified the minimum distances of sequences that are phylogenetically closest to the sequence id. Finally, the list of all closest sequences to the target sequences were extracted into a dictionary with the 171 target sequences as the keys and the closely related sequences are the values with their assigned species names specified in the output of all best matching subsequence, resulting in Table 5.



	id	closest_species	related_value	related_species
0	A115_pl__6__DL_14_4_2022_G4	Magneuptychia_ocypete	A115_pl__6__DL_14_4_2022_G4, A173_pl__6__DL_14...	Magneuptychia_ocypete, Magneuptychia_ocypete
1	A173_pl__6__DL_14_4_2022_F11	Magneuptychia_ocypete	A115_pl__6__DL_14_4_2022_G4, A173_pl__6__DL_14...	Magneuptychia_ocypete, Magneuptychia_ocypete
2	A222_pl__7__DL_14_4_2022_B7	Cissia_sp._NW108-6	A331_pl__6__DL_14_4_2022_H5, A322_pl__7__DL_14...	Cissia_myncea, Cissia_myncea, Cissia_myncea, C...
3	A272_pl__6__DL_14_4_2022_G12	Magneuptychia_ocypete	A115_pl__6__DL_14_4_2022_G4, A173_pl__6__DL_14...	Magneuptychia_ocypete, Magneuptychia_ocypete, ...
4	A273_pl__6__DL_14_4_2022_H1	Paryphthimoides_sylvina	A273_pl__6__DL_14_4_2022_H1, PM336_pl__7__DL_1...	Paryphthimoides_sylvina, Pierella_lamia
...	...	...	...	...
166	PM459_pl__8__DL_14_4_2022_D11	Hyalothyrus_sp._1YB	PM465_pl__8__DL_14_4_2022_E5	Hyalothyrus_sp._1YB
167	PM461_T7promoter.ab1	Salatis_sp._UK75	PM461_T7promoter.ab1, EC00022554-PM_469, EC000...	Salatis_sp._UK75, Calpodes_fusta, Calpodes_fus...
168	PM464_T7promoter.ab1	Euriphellus_phraxanorDHJ01	PM464_T7promoter.ab1, PM455_pl__8__DL_14_4_202...	Euriphellus_phraxanorDHJ01, Euriphellus_phraxa...
169	PM465_pl__8__DL_14_4_2022_E5	Hyalothyrus_sp._1YB	PM459_pl__8__DL_14_4_2022_D11	Hyalothyrus_sp._1YB
170	SAMPLE3_T7promoter.ab1	Mimoniades_ocyalus	A609_pl__2__DL_22_3_2022_F5	Mimoniades_ocyalus

171 rows × 4 columns

Table 5. Partial results of the identification of the closely related sequences within the phylogeny tree. For query sequences (ID) that had less than 95% p-ident values to the reference database, a tentative species identification is provided, which should be solid at the genus level or higher taxonomic rank depending on the reference database.

5 Discussion

Our semi-automated script was able to identify 1,555 sequences using a combination of BLAST searches and phylogenetic information using programming Python scripts and Bash. Our identifications are as robust as the reference database is, BOLD. However, because our query sequences come from a tropical locality, it was expected that several sequences were represented in databases, thus not able to identify their species identity. We quantified this gap as ~26% of species sampled during our field work (from 295 species, 78 did not have robust identifications with >95% similarity with any sequence in the reference database).

Our approach is scalable and only limited to computational resources available on desktop or local computers. In theory, our approach can also work offline, provided that the reference database is already constructed on local disks. This is important because it provides an alternative to online identification engines when new sequencing technologies allow researchers to sequence barcodes in the field, where internet connectivity might be a limitation.

This method works well because the solidly identified species (i.e., those with >95% percent identity) matched with butterfly species commonly encountered in the study location in Peru (pers. observations and compared to our local butterfly collection, which was taxonomically identified separately). In addition, we were able to easily spot suspicious sequences, such as *Wolbachia*, which is an endosymbiont commonly encountered in butterflies. Future possible expansions to this pipeline include a more automated program where end users state in the beginning of the approach what the level of percent identity is required (95% or more, depending on the organism) and the possibility to generate backups and updates in real-time with public databases and own sequences that were already identified, which can then be added safely to the reference database for future reference.

The project aimed to build a bioinformatic pipeline to semi-automatically identify the species sampled in the population in batch. This resulted in a dataset that can be used in future ecological and evolutionary studies to, for example, estimate phylogenetic diversity and species turnover across populations in the study region (in preparation). Additionally, the sequences can be used in future phylogenetic studies using the comparative method to understand the evolution of traits and whether there is any correlation of trait states with species diversification.

6 References

Altschul, Stephen F., Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. 1990. “Basic Local Alignment Search Tool.” *Journal of Molecular Biology* 215 (3): 403–10. https://doi.org/10.1016/S0022-2836(05)80360-2.

Cock, Peter J. A., Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, et al. 2009. “Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics.” *Bioinformatics (Oxford, England)* 25 (11): 1422–23. https://doi.org/10.1093/bioinformatics/btp163.

Hebert, Paul D. N., Alina Cywinska, Shelley L. Ball, and Jeremy R. deWaard. 2003. “Biological Identifications through DNA Barcodes.” *Proceedings of the Royal Society of London. Series B: Biological Sciences* 270 (1512): 313–21. https://doi.org/10.1098/rspb.2002.2218.

Katoh, Kazutaka, and Daron M. Standley. 2013. “MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability.” *Molecular Biology and Evolution* 30 (4): 772–80. https://doi.org/10.1093/molbev/mst010.

Kress, W. John, and David L. Erickson. 2008. “DNA Barcodes: Genes, Genomics, and Bioinformatics.” *Proceedings of the National Academy of Sciences* 105 (8): 2761–62. https://doi.org/10.1073/pnas.0800476105.

Matos-Maraví, Pável F., Carlos Peña, Keith R. Willmott, André V. L. Freitas, and Niklas Wahlberg. 2013. “Systematics and Evolutionary History of Butterflies in the ‘Taygetis Clade’ (Nymphalidae: Satyrinae: Euptychiina): Towards a Better Understanding of Neotropical Biogeography.” *Molecular Phylogenetics and Evolution* 66 (1): 54–68. https://doi.org/10.1016/j.ympev.2012.09.005.

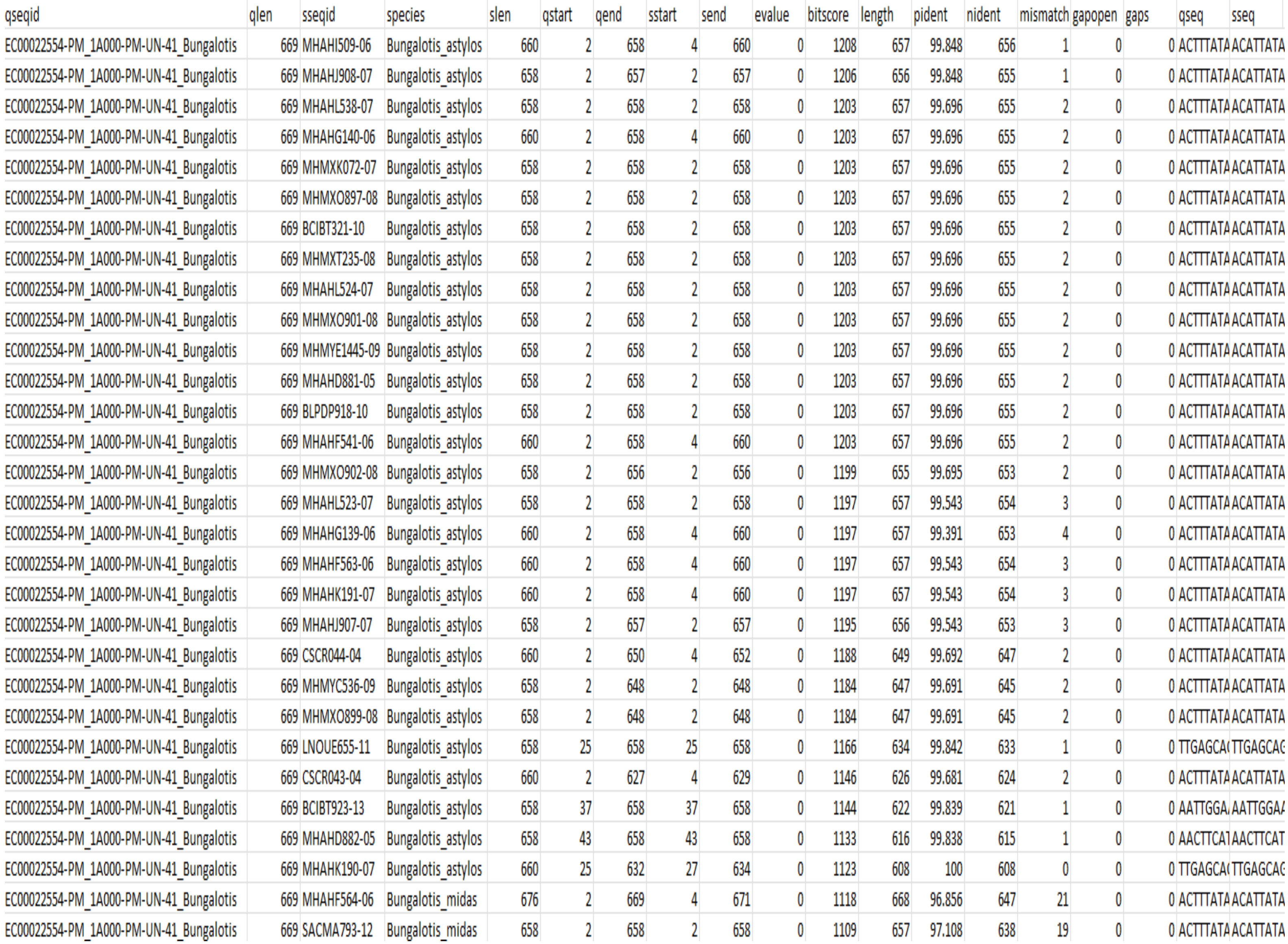
Minh, Bui Quang, Heiko A Schmidt, Olga Chernomor, Dominik Schrempf, Michael D Woodhams, Arndt von Haeseler, and Robert Lanfear. 2020. “IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era.” *Molecular Biology and Evolution* 37 (5): 1530–34. https://doi.org/10.1093/molbev/msaa015.

Nugent, Cam. (2019) 2019. “BOLD-CLI.” Go. https://github.com/CNuge/BOLD-CLI.

7 Tables

8 Appendix

Appendix table 1. The table displayed the top 30 headers from the BLAST of 1,555 sampled sequences.



qseqid	qlen	sseqid	species	slen	qstart	qend	sstart	send	evalue	bitscore	length	pident	nident	mismatch	gapopen	gaps	qseq	sseq
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHI509-06	Bungalotis_astylos	660	2	658	4	660	0	1208	657	99.848	656	1	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHJ908-07	Bungalotis_astylos	658	2	657	2	657	0	1206	656	99.848	655	1	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHL538-07	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHG140-06	Bungalotis_astylos	660	2	658	4	660	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXK072-07	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXO897-08	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	BCIBT321-10	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXT235-08	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHL524-07	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXO901-08	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMYE1445-09	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHD881-05	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	BLPDP918-10	Bungalotis_astylos	658	2	658	2	658	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHF541-06	Bungalotis_astylos	660	2	658	4	660	0	1203	657	99.696	655	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXO902-08	Bungalotis_astylos	658	2	656	2	656	0	1199	655	99.695	653	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHL523-07	Bungalotis_astylos	658	2	658	2	658	0	1197	657	99.543	654	3	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHG139-06	Bungalotis_astylos	660	2	658	4	660	0	1197	657	99.391	653	4	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHF563-06	Bungalotis_astylos	660	2	658	4	660	0	1197	657	99.543	654	3	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHK191-07	Bungalotis_astylos	660	2	658	4	660	0	1197	657	99.543	654	3	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHJ907-07	Bungalotis_astylos	658	2	657	2	657	0	1195	656	99.543	653	3	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	CSCR044-04	Bungalotis_astylos	660	2	650	4	652	0	1188	649	99.692	647	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMYC536-09	Bungalotis_astylos	658	2	648	2	648	0	1184	647	99.691	645	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHMXO899-08	Bungalotis_astylos	658	2	648	2	648	0	1184	647	99.691	645	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	LNOUE655-11	Bungalotis_astylos	658	25	658	25	658	0	1166	634	99.842	633	1	0	0	TTGAGCA	TTGAGCAG
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	CSCR043-04	Bungalotis_astylos	660	2	627	4	629	0	1146	626	99.681	624	2	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	BCIBT923-13	Bungalotis_astylos	658	37	658	37	658	0	1144	622	99.839	621	1	0	0	AATTGGA	AATTGGAA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHD882-05	Bungalotis_astylos	658	43	658	43	658	0	1133	616	99.838	615	1	0	0	AACTTCAT	AACTTCAT
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHK190-07	Bungalotis_astylos	660	25	632	27	634	0	1123	608	100	608	0	0	0	TTGAGCA	TTGAGCAG
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	MHAHF564-06	Bungalotis_midas	676	2	669	4	671	0	1118	668	96.856	647	21	0	0	ACTTTATA	ACATTATA
EC00022554-PM_1A000-PM-UN-41_Bungalotis	669	SACMA793-12	Bungalotis_midas	658	2	658	2	658	0	1109	657	97.108	638	19	0	0	ACTTTATA	ACATTATA

]

Appendix table 2.showing the total number of each specified species

Species	Samples_size
Achlyodes_busirus	7
Aethilla_echina	4
Agara_michaeli	1
Agara_perissodora	6
Aguna_asander	3
Aguna_claxon	9
Aguna_coeloides	1
Aides_brino	1
Aides_duma	4
Anartia_amathea	1
Anastrus_sp._UK47	1
Antigonus_erosus	3
Antigonus_nearchus	4
Apodemia_murphyi	1
Aroma_aroma	1
Astraptes_aulus	16
Astraptes_enotrus	1
Astraptes_janeiraDHJ01	1
Astraptes_janeiraDHJ02	1
Astraptes_mabillei	2
Augiades_crinisus	1
Autochton_Burns01DHJ02	1
Autochton_Burns01DHJ04	4
Autochton_longipennis	124
Autochton_zarex	17
Bia	1

Brachyglenis_dinora	1
Bungalotis_astylos	10
Cabares_potrillo	1
Calpodes_Burns08	2
Calpodes_antoninus	2
Calpodes_fusta	8
Calpodes_longirostrisDHJ02	3
Calpodes_severus	8
Carystoides_escalanteiDHJ02	2
Carystoides_escalanteiDHJ03	3
Carystoides_orbius	1
Carystus_phorcus	1
Cecropterus_dorantes	94
Cecropterus_doryssusDHJ02	24
Cecropterus_doryssus_doryssus	6
Cecropterus_jalapus	2
Cecropterus_virescens	2
Cecropterus_zarex	4
Celaenorrhinus_Burns03	1
Celaenorrhinus_approximatus	1
Celaenorrhinus_eligius	2
Celaenorrhinus_plagiatus	3
Cephise_Burns01	1
Ceratinia_tutia_poecila	1
Cercyonis_oetus	1
Chalypyge_chalybea	1
Chioides_catillus	116
Chloreuptychia_agatha	1

Chrysoplectrum_perniciosus	2
Cissia_myncea	3
Cissia_penelope	16
Cissia_proba	2
Cissia_sp._NW108-6	1
Cobalopsis_nero	2
Cobalus_virbius	1
Codatractus_alcaeus	1
Cogia_calchas	13
Cogia_goya	2
Cogia_optica	2
Cogia_stylites	15
Cogia_undulatus	17
Colobura_dirce	2
Crocozona_cf._coecias	1
Crocozona_coecias_coecias	1
Cyclosemia_Burns01	3
Cyclosemia_subcaerulea	1
Cymaenes_alumna	1
Cynea_Burns02	1
Cynea_cannae	10
Damas_clavus	2
Dircenna_loreta_melini	1
Dodona_elvira	3
Drephalys_kidonoi	21
Dryas_iulia	3
Dubiella_belpa	1
Dynamine_mexicanaDHJ02	1

Dyscophellus	4
Dyscophellus_porcius	1
Eantis_thraso	7
Ectima_thecla	1
Ectomis_Burns01	6
Ectomis_auginus	3
Ectomis_kanshul	2
Ectomis_orpheus	7
Elbella_adonis	4
Entheus_Burns03	1
Entheus_aureanota	2
Entheus_priassus_priassus	2
Epargyreus_Burns06	3
Epargyreus_cruza	59
Epargyreus_tmolis	1
Eresia_eunice	1
Eueides_evaDHJ01	2
Eueides_isabella	1
Euriphellus_euribates	1
Euriphellus_phraxanorDHJ01	3
Eurybia_elvina	2
Evansiella_cordela	1
Forbestra_olivencia	1
Godyris_zavaleta	1
Gorgopas_trochilus	2
Gufa_gulala	1
Hamadryas_epinome	1
Hedone_vibex	2

Heliconius_burneyi	2
Heliconius_elevatus	1
Heliconius_erato	1
Heliconius_erato_emma	4
Heliconius_erato_favorinus	4
Heliconius_ethilla	2
Heliconius_melpomene_aglaope	1
Heliconius_melpomene_amaryllis	2
Heliconius_melpomene_malleti	2
Heliconius_numata	1
Heliconius_numata_bicoloratus	1
Heliconius_numata_silvana	6
Heliconius_pardalinus_sergestus	3
Heliconius_sara	1
Hesperia_leonardus	1
Hoodus_pelopidas	1
Hyalothyrus_neleus	2
Hyalothyrus_sp._1YB	2
Hypanartia_lethe	1
Ithomia_salapia_aquinia	1
Jemadia_hewitsonii	1
Justinia_sp._UK57	1
Laparus_doris	3
Magneuptychia_libye	12
Magneuptychia_ocypete	7
Mechanitis_lysimnia	1
Melanis_sanguinea	1
Metardaris_cosinga	2

Metron_chrysogastra	1
Microceris_merops	2
Microceris_patrobasDHJ01	1
Microceris_patrobasDHJ05	11
Microceris_scylla	1
Mimoniades_ocyalus	4
Minasicles_vopiscus	4
Molo_mango	3
Morys_compta	2
Morys_geisa	1
Mylon_maimon	2
Mylothris_schumanni_uniformis	1
Mysoria_sejanus	1
Naevolus_orius	1
Narcosius_colossus	9
Nascus_Burns02	1
Nascus_paulliniae	2
Neoxeniades_Burns02	2
Neoxeniades_molion	2
Niconiades_gladys	3
Niconiades_incomptus	1
Nisoniades_Burns02	3
Nisoniades_bromias	1
Nisoniades_castolus	1
Nosphistia_zonara	2
Nyctelius_nycteliusDHJ01	2
Oeneis_ammon	1
Orphe_gerasa	2

Orses_cynisca	3
Ouleus_fridericus	1
Oxynthes_martius	2
Paches_loxus	2
Panoquina_ocola	6
Paracarystus_hypargira	1
Parelbella_macleannani	3
Paryphthimoides_sylvina	1
Perichares_adela	2
Perichares_philetes	7
Perichares_poaceaphaga	2
Perichares_prestoeaphaga	1
Phanus_vitreus	1
Phanus_vitreusDHJ01	6
Phareas_burnsi	2
Phocides_Burns01	2
Phocides_pigmalionDHJ02	1
Phocides_polybius	1
Phoebis_statira	1
Pierella_lamia	1
Polites_otho	1
Polythrix_sp._1YB	1
Pompeius_pompeius	1
Propertius_propertius	1
Pseudargynnis_hegemone	1
Pseudodebis_valentina	1
Pseudonascus_paulliniae	2
Pyrgus_malvae	7

Pyrgus_orcus	1
Pyrrhogyra_sp._1YB	1
Pyrrhopyge_phidias	6
Pythonides_amaryllis	1
Pythonides_proxenus	3
Quadrus_cerialis	1
Quadrus_contubernalis	2
Quasimellana_inconspicua	1
Rhetus_periander	1
Salatis_sp._UK75	1
Sarmientoia_similis	1
Sodalia_coler	1
Spathilepia_clonius	5
Spicauda_procne	13
Spicauda_simplicius	220
Spicauda_tanna	16
Spicauda_teleus	57
Spioniades_artemides	1
Staphylus_Janzen03	1
Staphylus_caribbea	1
Staphylus_melangon	1
Talides_Burns02	1
Tarsoctenus_corytus	1
Taygetis_virgilia	1
Telegonus_SENNOVnumt	8
Telegonus_alardus	7
Telegonus_anaphus	6
Telegonus_anausis_annettaDHJ02	4

Telegonus_anausis_annettaDHJ03	2
Telegonus_azul	1
Telegonus_chiriquensis	1
Telegonus_creteus_cranaDHJ01	5
Telegonus_favilla	2
Telegonus_fruticibus	9
Telegonus_fulgerator	1
Telegonus_fulminator	5
Telegonus_hopfferiDHJ02	2
Telegonus_obstupefactus	8
Telegonus_procrastinator	1
Telegonus_synecdoche	7
Telegonus_synecdochenumt	22
Telemiades_Burns08	1
Telemiades_fides	1
Telemiades_lamasi	1
Telemiades_meris	3
Thoon_ponka	1
Thracides_cleanthes	4
Thracides_nanea_nida	1
Thracides_phidon	4
Tithorea_harmonia	3
Toxidia_andersoni	1
Toxidia_rietmanni	1
Turesis_complanula	2
Urbanus_albimargo	2
Urbanus_alva	8
Urbanus_esmeraldus	48

Urbanus_esta	9
Urbanus_parvus	5
Urbanus_pronta	2
Urbanus_proteus	13
Urbanus_segnestami	21
Vacerra_aeasDHJ01	5
Vehilius_vetula	1
Vertica_subrufescensDHJ02	1
Vettius_artona	1
Vettius_marcus	1
Vettius_picaDHJ02	1
Xeniades_orchamus	2
Xenophanes_tryxus	3
Yanguna_thelersa	4
Yphthimoides_renata	8
Zariaspes_mys	2
Zemeros_flegyas	1

Appendix table 3. The table showed total number of 217 identified unique species with confident similarity rate ≥ 95%

Species	n (samples)
Achlyodes_busirus	7
Aethilla_echina	3
Agara_michaeli	1
Agara_perissodora	6
Aguna_claxon	9
Aguna_coeloides	1
Aides_brino	1
Aides_duma	4
Anartia_amathea	1
Anastrus_sp._UK47	1

Antigonus_erosus	3
Antigonus_nearchus	4
Aroma_aroma	1
Astraptes_aulus	16
Astraptes_enotrus	1
Astraptes_janeiraDHJ01	1
Astraptes_janeiraDHJ02	1
Astraptes_mabillei	1
Augiades_crinisus	1
Autochton_Burns01DHJ02	1
Autochton_Burns01DHJ04	4
Autochton_longipennis	118
Autochton_zarex	17
Bia	1
Bungalotis_astylos	10
Calpodes_Burns08	2
Calpodes_antoninus	2
Calpodes_fusta	8
Calpodes_longirostrisDHJ02	3
Calpodes_severus	8
Carystus_phorcus	1
Cecropterus_dorantes	94
Cecropterus_doryssusDHJ02	24
Cecropterus_doryssus_doryssus	6
Cecropterus_jalapus	2
Cecropterus_virescens	2
Cecropterus_zarex	2
Celaenorrhinus_eligius	1
Ceratinia_tutia_poecila	1
Chioides_catillus	96
Chloreuptychia_agatha	1
Chrysoplectrum_perniciosus	2
Cissia_myncea	3
Cissia_penelope	16
Cissia_proba	2
Cobalopsis_nero	2
Cobalus_virbius	1
Codatractus_alcaeus	1
Cogia_calchas	11
Cogia_goya	1
Cogia_stylites	6

Cogia_undulatus	11
Colobura_dirce	2
Crocozona_cf._coecias	1
Crocozona_coecias_coecias	1
Cymaenes_alumna	1
Cynea_Burns02	1
Cynea_cannae	10
Dircenna_loreta_melini	1
Drephalys_kidonoi	20
Dryas_iulia	3
Dynamine_mexicanaDHJ02	1
Dyscophellus	4
Dyscophellus_porcius	1
Eantis_thraso	7
Ectima_thecla	1
Ectomis_Burns01	6
Ectomis_auginus	3
Ectomis_orpheus	5
Elbella_adonis	3
Entheus_aureanota	2
Entheus_priassus_priassus	2
Epargyreus_Burns06	2
Epargyreus_cruza	59
Epargyreus_tmolis	1
Eresia_eunice	1
Eueides_evaDHJ01	2
Eueides_isabella	1
Euriphellus_euribates	1
Eurybia_elvina	2
Forbestra_olivencia	1
Godyris_zavaleta	1
Gorgopas_trochilus	2
Hamadryas_epinome	1
Hedone_vibex	2
Heliconius_burneyi	2
Heliconius_elevatus	1
Heliconius_erato	1
Heliconius_erato_emma	4
Heliconius_erato_favorinus	4
Heliconius_ethilla	2
Heliconius_melpomene_aglaope	1

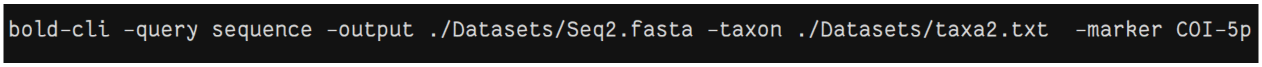
Heliconius_melpomene_amaryllis	2
Heliconius_melpomene_malleti	2
Heliconius_numata	1
Heliconius_numata_bicoloratus	1
Heliconius_numata_silvana	6
Heliconius_pardalinus_sergestus	3
Heliconius_sara	1
Hyalothyrus_neleus	2
Hypanartia_lethe	1
Ithomia_salapia_aquinia	1
Jemadia_hewitsonii	1
Justinia_sp._UK57	1
Laparus_doris	3
Magneuptychia_libye	12
Magneuptychia_ocypete	4
Mechanitis_lysimnia	1
Melanis_sanguinea	1
Metardaris_cosinga	2
Metron_chrysogastra	1
Microceris_merops	2
Microceris_patrobasDHJ05	1
Microceris_scylla	1
Minasicles_vopiscus	4
Molo_mango	3
Morys_compta	2
Morys_geisa	1
Mylon_maimon	2
Naevolus_orius	1
Narcosius_colossus	7
Nascus_Burns02	1
Nascus_paulliniae	2
Neoxeniades_Burns02	2
Neoxeniades_molion	2
Niconiades_gladys	3
Niconiades_incomptus	1
Nisoniades_castolus	1
Nosphistia_zonara	2
Nyctelius_nycteliusDHJ01	2
Orphe_gerasa	2
Orses_cynisca	3
Ouleus_fridericus	1

Oxynthes_martius	2
Paches_loxus	2
Panoquina_ocola	6
Paracarystus_hypargira	1
Parelbella_macleannani	2
Perichares_adela	2
Perichares_philetes	7
Perichares_poaceaphaga	2
Perichares_prestoeaphaga	1
Phanus_vitreus	1
Phanus_vitreusDHJ01	6
Phareas_burnsi	1
Phocides_Burns01	2
Phocides_pigmalionDHJ02	1
Phocides_polybius	1
Phoebis_statira	1
Pierella_lamia	1
Polites_otho	1
Polythrix_sp._1YB	1
Pompeius_pompeius	1
Pseudodebis_valentina	1
Pseudonascus_paulliniae	2
Pyrgus_orcus	1
Pyrrhogyra_sp._1YB	1
Pyrrhopyge_phidias	6
Pythonides_proxenus	1
Quadrus_cerialis	1
Quadrus_contubernalis	2
Quasimellana_inconspicua	1
Sarmientoia_similis	1
Sodalia_coler	1
Spathilepia_clonius	5
Spicauda_procne	13
Spicauda_simplicius	215
Spicauda_tanna	16
Spicauda_teleus	57
Spioniades_artemides	1
Staphylus_Janzen03	1
Staphylus_melangon	1
Talides_Burns02	1
Taygetis_virgilia	1

Telegonus_SENNOVnumt	8
Telegonus_alardus	7
Telegonus_anaphus	6
Telegonus_anausis_annettaDHJ02	4
Telegonus_anausis_annettaDHJ03	2
Telegonus_azul	1
Telegonus_chiriquensis	1
Telegonus_creteus_cranaDHJ01	5
Telegonus_favilla	2
Telegonus_fruticibus	7
Telegonus_fulgerator	1
Telegonus_fulminator	5
Telegonus_hopfferiDHJ02	2
Telegonus_obstupefactus	8
Telegonus_synecdoche	7
Telegonus_synecdochenumt	17
Telemiades_Burns08	1
Telemiades_fides	1
Telemiades_meris	3
Thoon_ponka	1
Thracides_cleanthes	4
Thracides_phidon	1
Tithorea_harmonia	3
Turesis_complanula	2
Urbanus_albimargo	2
Urbanus_alva	7
Urbanus_esmeraldus	48
Urbanus_esta	9
Urbanus_parvus	5
Urbanus_pronta	2
Urbanus_proteus	13
Urbanus_segnestami	19
Vacerra_aeasDHJ01	5
Vehilius_vetula	1
Vertica_subrufescensDHJ02	1
Vettius_artona	1
Vettius_marcus	1
Vettius_picaDHJ02	1
Xeniades_orchamus	2
Xenophanes_tryxus	3
Yanguna_thelersa	3

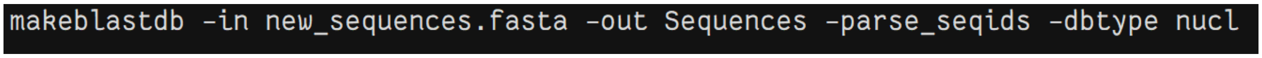
Yphthimoides_renata	8
Zariaspes_mys	1

Appendix code 1.BOLD-CLI command to retrive the databases on BOLD Systems for butterflies.



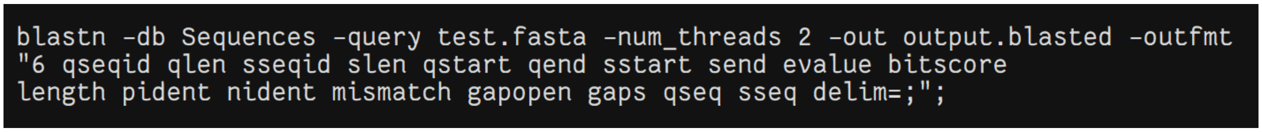
```
bold-cli -query sequence -output ./Datasets/Seq2.fasta -taxon ./Datasets/taxa2.txt  -marker COI-5p
```

Appendix code 2.The makeblastdb command to create databases from the metadatabases.



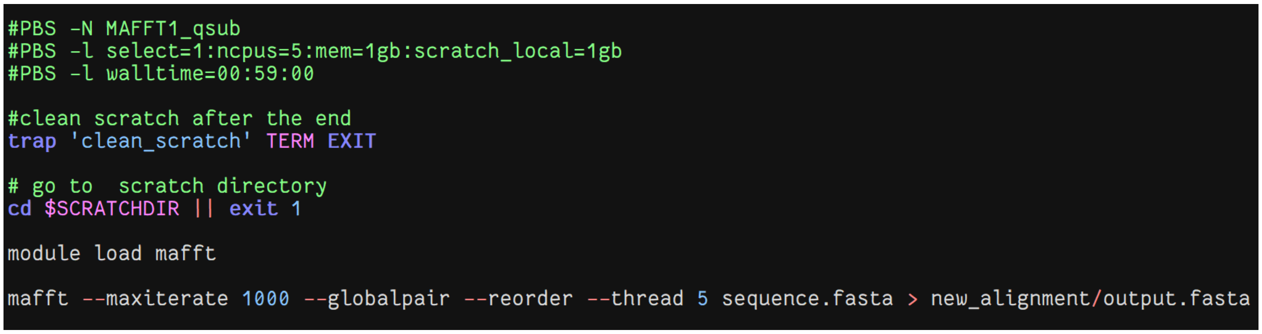
```
makeblastdb -in new_sequences.fasta -out Sequences -parse_seqids -dbtype nucl
```

Appendix code 3.The blastn command to query the best matches bettwween our databases and the output from makeblastdb command.



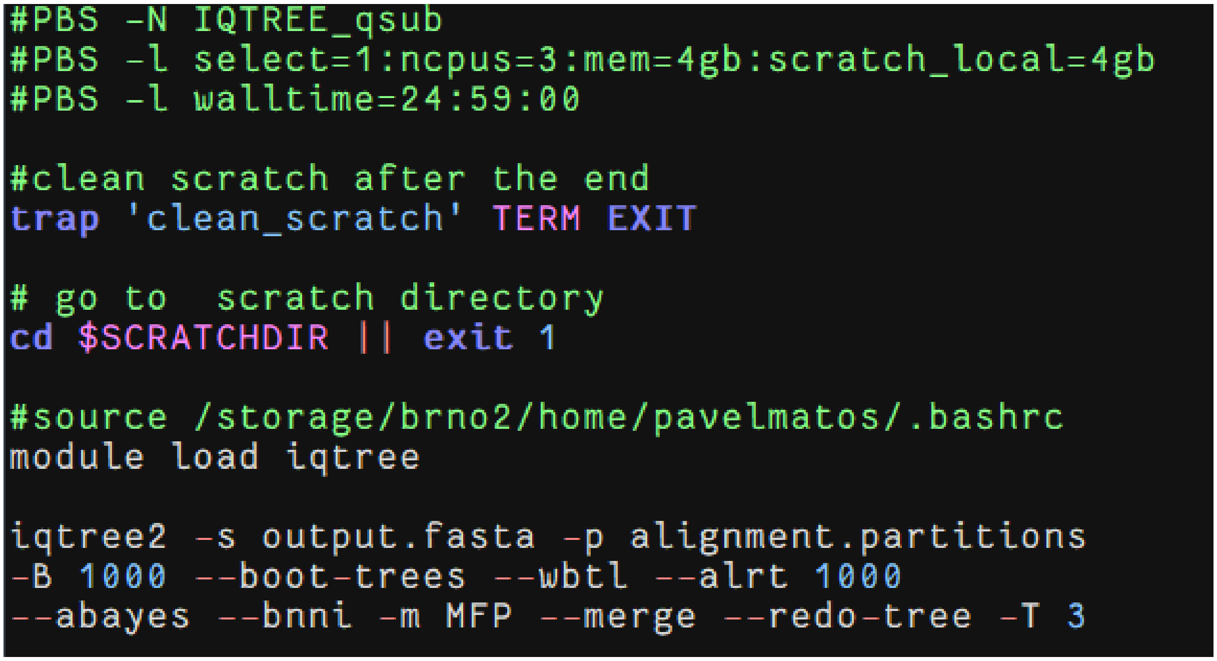
```
blastn -db Sequences -query test.fasta -num_threads 2 -out output.blasted -outfmt
"6 qseqid qlen sseqid slen qstart qend sstart send evalue bitscore
length pident nident mismatch gapopen gaps qseq sseq delim=;";
```

Appendix code 4.The script of mafft to submit on metacentrum to align the obtained COI sequences.



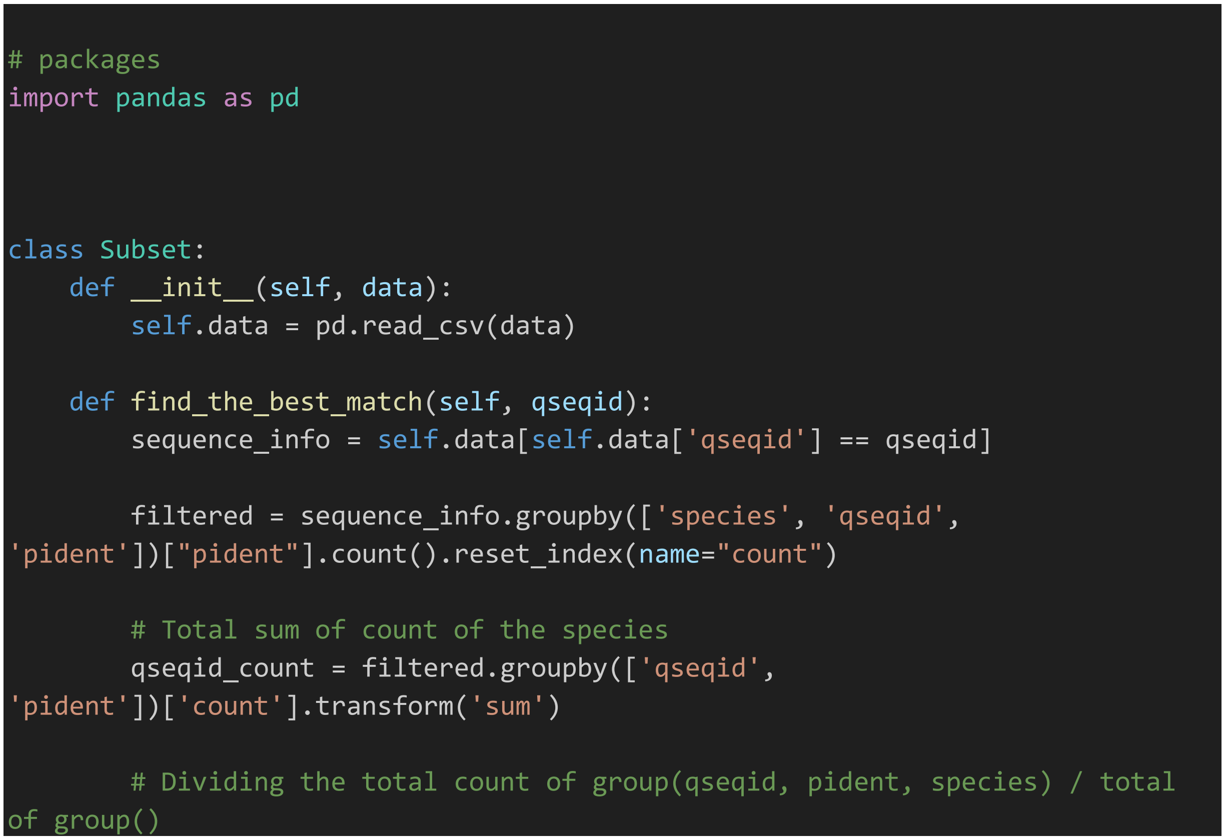
```
#PBS -N MAFFT1_qsub
#PBS -l select=1:ncpus=5:mem=1gb:scratch_local=1gb
#PBS -l walltime=00:59:00

#clean scratch after the end
trap 'clean_scratch' TERM EXIT

# go to  scratch directory
cd $SCRATCHDIR || exit 1

module load mafft

mafft --maxiterate 1000 --globalpair --reorder --thread 5 sequence.fasta > new_alignment/output.fasta
```

Appendix code 5.The script of iqtree2 to submit on Metacentrum to construct the phylogeny tree.



```
#PBS -N IQTREE_qsub
#PBS -l select=1:ncpus=3:mem=4gb:scratch_local=4gb
#PBS -l walltime=24:59:00

#clean scratch after the end
trap 'clean_scratch' TERM EXIT

# go to  scratch directory
cd $SCRATCHDIR || exit 1

#source /storage/brno2/home/pavelmatos/.bashrc
module load iqtree

iqtree2 -s output.fasta -p alignment.partitions
-B 1000 --boot-trees --wbtl --alrt 1000
--abayes --bnni -m MFP --merge --redo-tree -T 3
```

Appendix code 6. Python script to retrieve the best matches of the sequences



```
# packages
import pandas as pd



class Subset:
    def __init__(self, data):
        self.data = pd.read_csv(data)

    def find_the_best_match(self, qseqid):
        sequence_info = self.data[self.data['qseqid'] == qseqid]

        filtered = sequence_info.groupby(['species', 'qseqid',
'pident'])["pident"].count().reset_index(name="count")

        # Total sum of count of the species
        qseqid_count = filtered.groupby(['qseqid',
'pident'])['count'].transform('sum')

        # Dividing the total count of group(qseqid, pident, species) / total
of group()
```

```
        filtered['frequency'] = round((filtered['count'] / qseqid_count) *
100, 4)

        max_freq_row = filtered.loc[filtered.groupby(['qseqid',
'pident'])['count'].idxmax()]

        # Find the row with the maximum pident within each species
        max_pident_row =
max_freq_row.loc[max_freq_row.groupby('qseqid')['pident'].idxmax()]

        return max_pident_row

    def best_of_subset(self):
        lst = []
        for qseqid in self.data['qseqid'].unique():
            result = self.find_the_best_match(qseqid)
            lst.append(result)
        result_df = pd.concat(lst, ignore_index=True)
        return result_df
#Excecuting the class and converting it into dataframe
data= 'output_latest_new.csv'
df = Subset(data)
result_df = df.best_of_subset()

result_df.to_csv('latest_blast_2.csv')


#Convert datasets with p-ident >= 95%

blast_95 = result_df[result_df['pident'] >= 95]
blast_95.to_csv('blast_95.csv')

#Convert datasets with p-ident <95%
blast = result_df[result_df['pident'] < 95]
blast.to_csv('blast.csv')
```

Appendix code 7. Python script to find the missing sequences in the ‘best_of_subset’ data

```
def filter_not_in(data,names):
    df = pd.read_csv(data,index_col=0)
    names = pd.read_csv(names)
    df_qid= df['qseqid'].tolist()
    qid =names['id'].str.lstrip(">").tolist()


#     Initial lst of names are not in the lst of dataframe
    not_in_lst=[]

    for id1 in qid:
        if id1 not in df_qid:
            not_in_lst.append(id1)
    table = pd.DataFrame({'id':not_in_lst })
    return table
```

Appendix code 8. Python script to find the closely related sequences

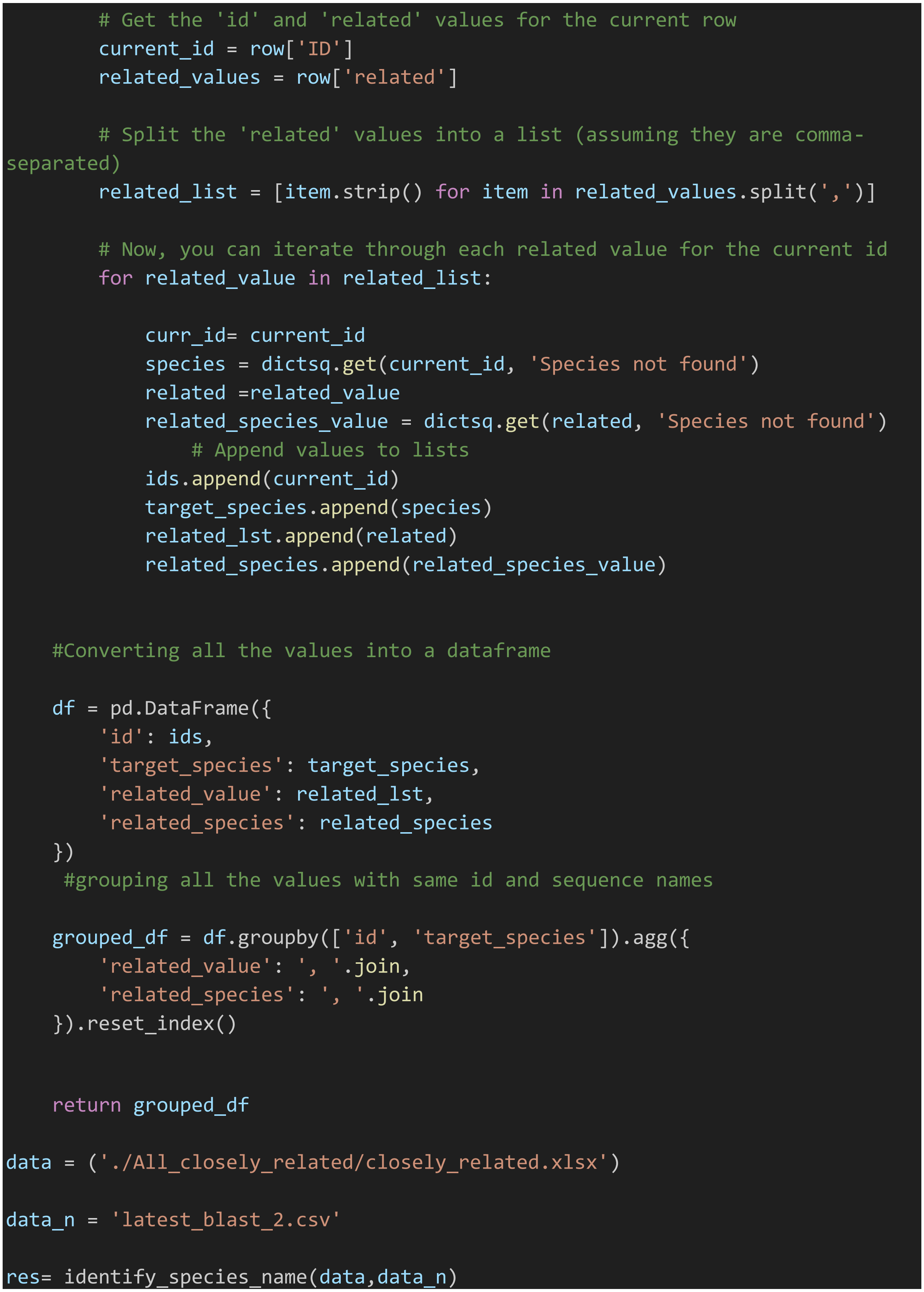
```
from tqdm import tqdm
import time
from Bio import Phylo

#run with gpu

class FindingCloselyRelated:
    def __init__(self, lst, tree, device=('cuda' if torch.cuda.is_available()
else 'cpu'):
        self.lst = lst
        self.tree = Phylo.read(tree, "newick")  # Read the tree here
        self.device = device

    def get_closest_taxa(self, target_species):
        closest_taxa = None
        min_distance = float('inf')

        for clade in self.tree.find_clades():
            if clade.name == target_species:
                continue

            # Convert to torch tensors for computation
            target_tensor = torch.tensor(self.tree.distance(target_species,
clade.name), device=self.device)
```

```
            # Update closest taxa if the current taxon is closer
            if target_tensor < min_distance:
                min_distance = target_tensor.item()
                closest_taxa = [(clade.name, clade, [tip.name for tip in
clade.get_terminals()])]
            elif target_tensor == min_distance:
                closest_taxa.append((clade.name, clade, [tip.name for tip in
clade.get_terminals()]))

        return closest_taxa, min_distance

    def find_target_lst(self):
        dicts = {}
        total_species = len(self.lst)
        total_elapsed_time = 0.0

        # Use tqdm for progress tracking
        for target_species in tqdm(self.lst, desc='Processing',
total=total_species):
            start_time = time.time()
            with torch.cuda.device(self.device):
                closest_taxa, min_distance =
self.get_closest_taxa(target_species)
            elapsed_time = time.time() - start_time
            total_elapsed_time += elapsed_time

            for taxon_id, taxon_name, tip_labels in closest_taxa:
                dicts[target_species] = dicts.get(target_species, [',
'.join(tip_labels)])

            tqdm.write(f"Processed {target_species} in {elapsed_time:.2f}
seconds")

        tqdm.write(f"Total processing time: {total_elapsed_time:.2f} seconds")

        return dicts



#Excuting the class FindingCloselyRelated

data = pd.read_csv('blast.csv')
species_list = list(data['qseqid'])  # replace with your actual species list
tree_file = "alignment.partitions.treefile"  # replace with your actual tree
file
```

```
finder = FindingCloselyRelated(species_list, tree_file)
result_dicts = finder.find_target_lst()


#Convert the result into dataframe
df = pd.DataFrame(
    [(k, val) for k, vals in result_dicts.items() for val in vals],
    columns=['ID', 'related']
)


df.to_excel('closely_related.xlsx')
```

Appendix code 9. Python script to retrieve the species names for the target sequences and its list of all closely related sequences.

```
def identify_species_name(data_xlsx, data2_csv):
    # the output from finding closely related from phylogeny tree
    data = pd.read_excel(data_xlsx)


    # The best matching subsequences file
    data2= pd.read_csv(data2_csv, index_col=0)

    #Initialize the empy dictionary to create new dictionary with qseqid and
species names
    dictsq= {}
    for values in data2.values:
        qseq = values[1]
        spec = values[0]
        dictsq[qseq] = dictsq.get(qseq,spec )


    ids = []
    target_species = []
    related_lst = []
    related_species = []




    # Iterate through each row in the DataFrame
    for index, row in data.iterrows():
```



```
        # Get the 'id' and 'related' values for the current row
        current_id = row['ID']
        related_values = row['related']

        # Split the 'related' values into a list (assuming they are comma-
separated)
        related_list = [item.strip() for item in related_values.split(',')]

        # Now, you can iterate through each related value for the current id
        for related_value in related_list:

            curr_id= current_id
            species = dictsq.get(current_id, 'Species not found')
            related =related_value
            related_species_value = dictsq.get(related, 'Species not found')
                # Append values to lists
            ids.append(current_id)
            target_species.append(species)
            related_lst.append(related)
            related_species.append(related_species_value)


    #Converting all the values into a dataframe

    df = pd.DataFrame({
        'id': ids,
        'target_species': target_species,
        'related_value': related_lst,
        'related_species': related_species
    })
     #grouping all the values with same id and sequence names

    grouped_df = df.groupby(['id', 'target_species']).agg({
        'related_value': ', '.join,
        'related_species': ', '.join
    }).reset_index()


    return grouped_df

data = ('./All_closely_related/closely_related.xlsx')

data_n = 'latest_blast_2.csv'

res= identify_species_name(data,data_n)
```

```
#Saving the dataframe into the csv file

res.to_csv('attached_species_names.csv')
```