SuperCRUNCH: A toolkit for creating and manipulating supermatrices and other large phylogenetic datasets

Daniel M. Portik[1*], John J. Wiens[1]

1. Department of Ecology and Evolutionary Biology, University of Arizona, Tucson, Arizona

*Correspondence: danielportik@email.arizona.edu, daniel.portik@gmail.com

Running Title: SuperCRUNCH for phylogenetic data

Keywords: bioinformatics, GenBank, genomics, multiple sequence alignment, orthology, phylogenetics, phylogeography, supermatrix

Abstract

1. Phylogenies with extensive taxon sampling have become indispensable for many types of ecological and evolutionary studies. Many large-scale trees are based on a "supermatrix" approach, which involves amalgamating thousands of published sequences for a group. Constructing up-to-date supermatrices can be challenging, especially as new sequences may become available within the group almost constantly. However, few tools exist for assembling large-scale, high-quality supermatrices (and other large datasets) for phylogenetic analysis.

2. Here we present SuperCRUNCH, a Python toolkit for assembling large phylogenetic datasets from GenBank/NCBI nucleotide data. SuperCRUNCH searches for specified sets of taxa and loci to create species-level or population-level datasets. It offers many transparent options for orthology detection, sequence selection, alignment, and file manipulation for generating large-scale phylogenetic datasets.

3. We compared SuperCRUNCH to the most recent alternative approach for generating supermatrices (PyPHLAWD) for two datasets. Given the same set of starting sequences, SuperCRUNCH required more computational time but it retrieved more taxa and total sequences and produced trees having greater congruence with previous studies. SuperCRUNCH can assemble supermatrices for genomic datasets with thousands of loci, and can also generate population-level datasets for phylogeographic analyses. We demonstrate clear advantages for using data downloaded directly from GenBank/NCBI rather than using intermediate databases. Furthermore, we show the effectiveness of initially identifying loci through label searching followed by rigorous orthology detection, rather than relying on automated clustering of all sequences. SuperCRUNCH is open-source, well-documented, and freely available at

https://github.com/dportik/SuperCRUNCH, with several complete example analyses available at https://osf.io/bpt94/.

4. SuperCRUNCH is a flexible method that can be used to assemble high quality phylogenetic datasets for any taxonomic group and for any scale (kingdoms to individuals). It allows rapid construction of supermatrices, greatly simplifying the process of updating large phylogenies with new data. SuperCRUNCH streamlines the major tasks required to process sequence data, including filtering, alignment, trimming, and formatting.

1 | INTRODUCTION

Large-scale phylogenies, including hundreds or thousands of species, have become essential for many studies in ecology and evolutionary biology. Many of these large-scale phylogenies are based on the supermatrix approach (e.g., de Queiroz & Gatesy, 2007), which typically involves amalgamating thousands of sequences from public databases (e.g., GenBank). Yet only a handful of tools exist for automatically assembling these large phylogenetic datasets. These include programs like PhyLoTA (Sanderson, Boss, Chen, Cranston, & Wehe 2008), PHLAWD (Smith, Beaulieu, & Donoghue 2009), phyloGenerator (Pearse & Purvis, 2013), SUMAC (Freyman, 2015), SUPERSMART (Antonelli et al., 2017), PhylotaR (Bennett et al., 2018) and PyPHLAWD (Smith & Walker, 2018). Each program has its own pros and cons for assembling molecular datasets. However, many of them rely on particular GenBank database releases to retrieve starting sequences (i.e., PhyLoTA, PyPHLAWD, SUMAC, SUPERSMART). GenBank releases occur every two months, and unless continually updated in the workflow, all new sequence data are automatically excluded from searches. Other workflows use the NCBI taxonomic database to locate sequence data, which can conflict with clade-specific taxonomies and inadvertently exclude relevant records. Many programs (e.g., PHLAWD, PyPHLAWD, PhyLoTA, PhylotaR, SUPERSMART) also employ automated (blind) clustering of all sequences, which produces orthologous sequence clusters that represent either individual or aggregate loci (such as longer mtDNA sequences spanning multiple genes). Although blind clustering may be useful under some circumstances, the ability to target specific loci instead may often be desirable. In addition, the criteria for orthology detection, filtration, and sequence selection are not always clear in these programs. Thus, producing high-quality phylogenetic datasets is presently challenging using

many of the available methods. because of the rapid influx of new sequence data, development of new loci, changing taxonomies, and lack of methodological transparency.

To address these challenges, we developed SuperCRUNCH, a semi-automated method for extracting, filtering, and manipulating nucleotide data. SuperCRUNCH uses sequence data downloaded directly from NCBI as a starting point, rather than retrieving sequences through an intermediate step (i.e. via a GenBank database release and/or NCBI Taxonomy). Sequence sets are created based on designated lists of taxa and loci, offering fine-control for targeted searches. SuperCRUNCH also includes refined methods for orthology detection and sequence selection. By offering the option to select representative sequences for taxa or retain all filtered sequences, SuperCRUNCH can be used to generate interspecific supermatrix datasets (one sequence per taxon per locus) or population-level datasets (multiple sequences per taxon per locus). It can also be used to assemble phylogenomic datasets with thousands of loci. SuperCRUNCH is modular in design, is intended to be transparent, objective and repeatable, and offers flexibility across all major steps in constructing phylogenetic datasets. SuperCRUNCH is open-source and freely available at https://github.com/dportik/SuperCRUNCH.

2 | WORKFLOW

SuperCRUNCH consists of a set of PYTHON (v2.7) modules that function as stand-alone command-line scripts. These modules can be downloaded and executed independently without the need to install SuperCRUNCH as a PYTHON package or library, making them easy to use and edit. Nevertheless, there are eight dependencies that should be installed prior to use. These include the BIOPYTHON package for PYTHON, and the following seven external dependencies: NCBI-BLAST+ (for BLASTN and MAKEBLASTDB; Altschul, Gish, Miller, Myers, & Lipman 1990;

Camacho et al., 2009), CD-HIT-EST (Li & Godzik, 2006), CLUSTAL-O (Sievers et al., 2011), MAFFT (Katoh, Misawa, Kuma, & Miyata 2002; Katoh & Standley, 2013), MUSCLE (Edgar, 2004), MACSE (Ranwez, Douzery, Cambon, Chantret, & Delsuc 2018), and TRIMAL (Capella-Gutiérrez, Silla-Martínez, & Gabaldón 2009). Installation instructions for these programs, a comprehensive user-guide, and detailed usage instructions for all modules can be found at https://github.com/dportik/SuperCRUNCH. We provide several complete analysis examples on the Open Science Framework SuperCRUNCH project page available at https://osf.io/bpt94. We illustrate the overall workflow in Figs. 1–5. We briefly outline the major steps in a typical analysis below.

2.1 | Starting material

The starting material required for running SuperCRUNCH consists of a set of NCBI nucleotide sequence records downloaded in fasta format (sequential or interleaved), a list of taxonomic names, and a list of loci and associated search terms. We recommend obtaining starting sequences from the NCBI nucleotide database (GenBank) using searches for relevant taxonomy terms or organism identifier codes. Various search filters can be used to reduce the amount of non-target records and reduce the size of the downloaded file. However, this is not a necessary step. For clades with many species, downloading all records directly may not be possible. For these groups, the results of multiple searches using key organism identifiers can be downloaded and combined into a single fasta file.

SuperCRUNCH requires a list of taxon names that is used to filter sequences. Lists of taxon names for large clades can be obtained through general databases, such as the NCBI Taxonomy Browser. Many groups also have dedicated taxonomic databases, such as the Reptile

Database (Uetz, Freed, & Hošek 2018), AmphibiaWeb (2019), and "Amphibian Species of the World" (Frost, 2019). These usually contain up-to-date taxonomies in a downloadable format. Taxon names can also be extracted directly from fasta files using *Fasta_Get_Taxa.py*, though this is mainly recommended for population-level data. The taxon list can contain a mix of species (binomial names) and subspecies (trinomial names), and subspecies can either be included or excluded from searches. With the latter option, the subspecies component is ignored.

SuperCRUNCH also requires a list of loci and associated search terms to initially identify the content of sequence records. For each locus included in the list, SuperCRUNCH will search for associated abbreviated names and longer locus labels in the sequence record. The choice of loci to include will inherently be group-specific. Surveys of phylogenetic and phylogeographic papers should be used to identify an appropriate marker set. We acknowledge that the best criteria to decide whether loci should be included remain unresolved, but relevant criteria include how complete the taxon sampling is within the group for that locus (e.g., including only loci present in 20% or more of the species). The success of finding loci using SuperCRUNCH depends on defining appropriate locus abbreviations and labels. We recommend searching for the locus on GenBank to identify common labeling or using gene databases such as GeneCards (Stelzer et al., 2016). There is no upper limit to the number of loci. Thus, SuperCRUNCH can be used to search for large genomic datasets available on the NCBI database, such as those from ultraconserved elements (UCEs) and other sequence-capture approaches.

2.2 | Taxon filtering and locus extraction

Sequence records are extracted from the starting sequence database based on the user-provided list of taxon names and list of locus/gene search terms. An initial search with the taxon names list

can be performed using *Taxa_Assessment.py* to identify records with valid and invalid taxon names. An optional data-cleaning step (*Rename_Merge.py*) can be used to rescue records containing invalid taxon names that would otherwise be excluded. Loci are searched for using *Parse_Loci.py*, which requires the user-supplied taxon list and loci list with search terms. For each locus included receiving search hits, a fasta file is produced with records containing valid taxon names that matched one or more of the locus-specific search terms.

2.3 | Orthology filtering

Two main methods can be used in SuperCRUNCH to identify orthologous sequences within locus-specific fasta files. Each uses BLAST searches to extract orthologous sequences, but they differ in whether input reference sequences are required or not. In the first method (*Cluster_Blast_Extract.py*), sequences are clustered based on similarity using CD-HIT-EST, and a BLAST database is constructed from the largest cluster. Sequences from all clusters (including the reference cluster) are blasted to the database using a BLAST algorithm selected by the user (blastn, megablast, or dc-megablast). For each sequence with a significant match, the coordinates of all hits (excluding self-hits) are merged. This merging action often results in a single continuous base pair interval. However, non-overlapping coordinates can also be produced (stretches of N's, gene duplications, etc.). Multiple options are provided for handling these cases. All query sequences with significant hits are extracted based on the resulting coordinates.

The second method (*Reference_Blast_Extract.py*) relies on a previously assembled reference sequence set to construct the BLAST database, to which all sequences from the locus-specific fasta file are blasted. This second method is particularly useful for sequence records that contain multiple loci (such as those from organellar genomes) or for loci that have been

sequenced for non-overlapping fragments (which can interfere with automated clustering). An optional contamination-filtering step is available (*Contamination_Filter.py*). This step excludes all sequences scoring >95% identity for at least 100 continuous base pairs to the reference sequences (e.g., human mitochondrial genes).

2.4 | Sequence quality and selection

To construct a supermatrix, a single sequence is typically used to represent each species for each gene. If multiple sequences for a given gene exist for a given species (e.g., because multiple individuals were sampled) and only a single terminal taxon per species is desired, then an objective strategy must be used for sequence selection. SuperCRUNCH allows multiple options for sequence selection using *Filter_Seqs_and_Species*.py. These include the simplest solution: sorting sequences by length and selecting the longest sequence ("length" method). Another approach is available for protein-coding loci, in which sequences are first sorted by length and then undergo translation tests ("translate" method). Here, the longest sequence is translated in all forward and reverse frames in an attempt to identify a correct reading frame. If none are found, then the next-longest sequence is examined. If no sequences in the set passes translation, the longest sequence is selected rather than excluding the taxon. The "randomize" feature can be used to select a sequence randomly from the set available for a taxon, which can be used to generate supermatrix permutations. For all selection options, sequences must meet a minimum base pair threshold set by the user. This will determine the smallest amount of data that can be included for a given marker for a given terminal taxon (although the optimal minimum is another unresolved issue for supermatrix phylogenetics). The *Filter_Seqs_and_Species.py* module creates fasta files with a single sequence per taxon for all loci, and also provides other key output

files for reproducibility and transparency. For each locus, this includes a BatchEntrez compatible list of all accession numbers from the input file, a per-species list of accession numbers, and a comprehensive summary of the representative sequence selected for each species (accession number, length, translation test results, and number of alternative sequences available). The outputs from *Filter_Seqs_and_Species.py* can be used to infer the total number of possible supermatrix combinations (based on the number of available alternative sequences per taxon per locus) using *Infer_Supermatrix_Combinations.py*. Following the selection of representative sequences, the *Make_Acc_Table.py* module can be used to generate a table of NCBI accession numbers for all taxa and loci.

The *Filter_Seqs_and_Species.py* was initially designed to filter and select one sequence per species. However, retaining intraspecific sequence sets may actually be desirable for other projects (e.g., phylogeography), and we therefore include this option. In this case, rather than selecting the highest quality sequence available to represent a species, all sequences passing the filtration method ("translate" or "length") are retained.

2.5 | Alignment

SuperCRUNCH includes two pre-alignment steps and several options for multiple sequence alignment. One pre-alignment step (*Adjust_Direction.py*) adjusts the direction of all sequences in each locus-specific fasta file in combination with MAFFT. This step produces unaligned fasta files with all sequences written in the correct orientation (thereby avoiding major pitfalls with aligners). Sequences can be aligned using *Align.py* with one of several popular aligners (MAFFT, MUSCLE, CLUSTAL-O) or with all aligners sequentially. For coding loci, the MACSE translation aligner is also available, which is capable of aligning coding sequences with respect to their

translation while allowing for multiple frameshifts or stop codons. To use this aligner, the *Coding_Translation_Tests.py* module can be used to identify the correct reading frame of sequences, adjust them to the first codon position, and ensure completion of the final codon. Although MACSE can be run on a single set of reliable sequences (e.g., only those that passed translation), it has an additional feature allowing the simultaneous alignment of a set of reliable sequences and a set of unreliable sequences (e.g., those that failed translation) using different parameters. The *Coding_Translation_Tests.py* module can be used to generate all the necessary input files to perform this type of simultaneous alignment using MACSE.

2.6 | Concatenation and other post-alignment tasks

SuperCRUNCH contains several tools for alignment trimming and file formatting. The sequence records present in the fasta files (containing original description lines) can be relabeled at any stage using *Relabel_Fasta.py*. This option allows records to be renamed by species labels (with an option to include subspecies), by accession number, or by species label and accession number. For each fasta file included, *Relabel_Fasta.py* also produces a corresponding file containing the taxon labels, accession numbers, and original descriptions for easy reference. The *Trim_Alignments.py* module can be used trim all alignments with TRIMAL using a chosen gap-threshold value, the "noallgaps" method, or both. Fasta files can be converted into other commonly used formats (nexus, phylip) using *Fasta_Convert.py*.

After relabeling, species-level alignment files can be concatenated using *Concatenation.py*. The *Concatenation.py* module allows input and output formats of fasta or phylip, and allows the user to select the symbol to represent missing data (-, N, ?). *Concatenation.py* produces a log file containing the number of loci present for each terminal

taxon (the sum of which is the total number of sequences in the supermatrix) and a data partitions file (containing the corresponding base pairs of loci in the alignment). The *Concatenation.py* module creates associative arrays from individual alignments and can efficiently concatenate thousands of loci for any number of taxa (e.g., large sequence-capture datasets). The resulting locus-specific alignment files and concatenated alignments can be used as input for numerous phylogenetic or phylogeographic programs, such as RAxML (Stamatakos, 2014).

3 | DEMONSTRATIONS AND COMPARISONS

To demonstrate the full use of SuperCRUNCH we assembled a *de novo* supermatrix for Iguania, a clade of squamate reptiles that contains ~1900 species in 14 families (Uetz et al., 2018). This clade includes chameleons, dragons, flying lizards, iguanas, anoles, and other well-known lizards. For starting material, we downloaded all available iguanian sequence data from the NCBI nucleotide database on November 30, 2018 using the organism identifier code for Iguania (txid8511[Organism:exp]). This produced a 13.23 GB fasta file that included 8,785,378 records. We searched for 68 loci (61 nuclear, 7 mitochondrial) that have been widely used in reptile phylogenetics or phylogeography (Townsend, Alegre, Kelley, Wiens, & Reeder 2008; Portik, Wood, Grismer, Stanley, & Jackman 2012; Pyron, Burbrink, & Wiens 2013) (Supplementary Table 1). We used a modified version of the February 2018 release of the Reptile Database (which does not contain subspecies) to cross-reference all taxonomic names. For this SuperCRUNCH analysis the detailed usage of all modules is provided at https://osf.io/vsu5k/.

To demonstrate the ability of SuperCRUNCH to handle large sequence-capture datasets, we assembled a UCE supermatrix using recently published data for the microhylid frog genus

Kaloula (Alexander et al., 2017). We downloaded sequence data from the NCBI nucleotide database using the search terms *Kaloula* and "ultra conserved element", which resulted in a 32MB fasta file containing 38,568 records. We obtained a taxon list directly from the fasta file using *Fasta_Get_Taxa.py*, which contained ten species and four subspecies labels. We created a general-use UCE locus file from the UCE 5k probe set, which was used to conduct searches (complete details at https://osf.io/5rey2/).

To demonstrate the ability of SuperCRUNCH to produce population-level datasets, we created datasets for two genera (*Callisaurus, Uma*) belonging to the iguanian family Phrynosomatidae. Population-level datasets from a single study can sometimes be obtained from the corresponding supplemental or archived data, but SuperCRUNCH can be used to identify and gather all relevant sequences from multiple sources. We chose these two examples because we knew in advance that relevant phylogeographic data were present across multiple studies. We downloaded sequence data from the NCBI nucleotide database using the search term Phrynosomatidae, which resulted in a 52MB fasta file containing 82,557 records. We obtained a taxon list directly from the fasta file, which was pruned to contents of each respective genus for separate searches. For *Callisaurus* we also included two closely related outgroups (*Cophosaurus texanus, Holbrookia maculata*). For the *Uma notata* complex, we used other *Uma* species as outgroups. We used the set of 68 iguanian markers for searches and for suitable loci we retained all filtered available sequences (complete details at https://osf.io/49wz3/ and https://osf.io/6bwhf/).

We compared the performance of SuperCRUNCH to PyPHLAWD (Smith & Walker, 2018), a recently developed semi-automated method for supermatrix construction. Importantly, Smith & Walker (2018) showed that PyPHLAWD outperforms other available methods for

supermatrix construction based on the number of taxa and sequences retrieved. PyPHLAWD retrieves sequences through an intermediate step (via GenBank database release), and offers an option to target specific loci by using preassembled sets of reference sequences (referred to as a "baited" analysis). To search for the same set of 68 loci for Iguania, we performed a "baited" analysis in PyPHLAWD v1.0 with baits consisting of validated sequence sets obtained from our main SuperCRUNCH run (details at https://osf.io/5u43a/). Taxonomy was corrected post-analysis to exclude subspecies. To ensure that any differences in method performance were not related to the starting sequences, we used the same Iguania sequence set obtained by PyPHLAWD (134,028 records) to perform a second SuperCRUNCH analysis (details at https://osf.io/95qwx/).

We further compared these methods by performing analyses for the plant clade Dipsacales. This was the example dataset used by Smith and Walker (2018). Following Smith and Walker (2018), we searched for the same four loci (ITS, matK, rbcl, trnL-trnF) using a "baited" analysis in PyPHLAWD v1.0 with their provided bait sets (details at https://osf.io/zpxbr/). We performed a SuperCRUNCH analysis using the Dipsacales starting sequence set obtained by PyPHLAWD (12,348 records), with both analyses including subspecies (details at https://osf.io/hk8db/).

For each of the six resulting supermatrices and all population-level loci recovered, we ran unpartitioned RAxML analyses (Stamatakis, 2014). We used the GTRCAT model and 100 rapid bootstraps to generate a phylogeny with support values for each dataset. To summarize sets of relationships within trees we developed an additional Python-based program called MonoPhylo. MonoPhylo assesses the status (monophyletic, paraphyletic, polyphyletic) of any number of user-defined groupings (genus, subfamily, family, etc.) for the tips present in a given tree. For

each grouping, MonoPhylo outputs the number of taxa defining the group, the status of the group, a support value if it is monophyletic (for bootstrapped trees), and if it is not monophyletic the number of interfering taxa and their corresponding tip labels. MonoPhylo relies on the Python ETEv3 tookit (Huerta-Cepas, Serra, & Bork 2016), and is open-source and freely available with detailed instructions at https://github.com/dportik/MonoPhylo.

All SuperCRUNCH and PyPHLAWD analyses were run on an iMac with a 4.2 GHz quad-core Intel Core i7 with 32 GB RAM. The materials required to replicate all analyses, including all input and output files, are freely available from the SuperCRUNCH project page on the Open Science Framework (https://osf.io/bpt94/).

4 | RESULTS

SuperCRUNCH generally outperformed PyPHLAWD in all comparisons and resulted in higher numbers of sequences and taxa for Iguania and Dipsacales (Table 1). For Iguania, the SuperCRUNCH analysis of data downloaded directly from NCBI resulted in the largest supermatrix, with 1,426 taxa, 66 loci, and 13,419 total sequences. This supermatrix was constructed from a set of 58,246 total filtered sequences which allowed for 4.2×10^{3155} possible supermatrix combinations. After controlling for differences in starting sequences, SuperCRUNCH still outperformed PyPHLAWD with respect to the number of taxa (1,358 vs. 1,069), loci (66 vs. 65), and sequences (12,654 vs. 10,397) included in the final supermatrix (Table 1). The analysis of Dipsacales was smaller in scope. Nevertheless, SuperCRUNCH still recovered more taxa (651 vs. 641) and sequences (1,598 vs. 1,510; Table 1, Supplementary Table 2).

A per-locus comparison for Iguania revealed the largest difference in performance between PyPHLAWD and SuperCRUNCH was for mtDNA loci. PyPHLAWD found only 38% of the total mtDNA sequences recovered by SuperCRUNCH (Supplemental Table 1) despite having access to an extensive bait set. PyPHLAWD recovered only four of the seven mitochondrial genes with reasonable success (>50 sequences). Results for the nuclear loci were more similar between methods, with both methods recovering equal results for 45 loci and with SuperCRUNCH obtaining better results for 11 loci (Supplemental Table 1). PyPHLAWD obtained more sequences for three nuclear loci, but close inspection revealed this was due to synonomous gene names for two loci that we were unaware of (R35 vs. GPR14, ZEB2 vs. ZHFX1b) and the inclusion of paralogous sequence data for one locus (ENC1 contained ENC6 sequences).

PyPHLAWD did outperform SuperCRUNCH by one criterion: total run times were faster for PyPHLAWD than for SuperCRUNCH (Table 1). However, the majority of the time required to run SuperCRUNCH involved the sequence adjustment and alignment steps (61–92% of the total run time; Supplementary Tables 3–8). Our reported times were calculated from a worst-case scenario of running each of the three to four aligners sequentially. In practice they can be run in parallel, which reduces the total time required. For the Iguania analysis running all four aligners, this parallel approach would have reduced the total run time by 47% (from ~10 hrs to ~5 hrs).

Phylogenetic trees estimated from the constructed supermatrices are available at https://osf.io/2twja/. For Iguania, we compared the full SuperCRUNCH phylogeny (1,426 taxa) and the PyPHLAWD phylogeny (1,069 taxa) by evaluating the number of genera, subfamilies, and families recovered as monophyletic using MonoPhylo (Supplemental Tables 9–10). The SuperCRUNCH phylogeny recovered all 14 families as monophyletic with high support (mean

bootstrap support: 95%), whereas PyPHLAWD recovered 13 of 14 families as monophyletic (mean bootstrap support: 98%). The PyPHLAWD analysis failed to support the monophyly of Agamidae (i.e., Chamaeleonidae is nested inside Agamidae), even though agamid monophyly has been strongly supported in previous supermatrix analyses (Pyron et al. 2013; Zheng & Wiens, 2016) and phylogenomic analyses (Streicher, Schulte, & Wiens 2016). Both SuperCRUNCH and PyPHLAWD recovered all 10 subfamilies as monophyletic (mean bootstrap support: 99% and 98%, respectively). The SuperCRUNCH phylogeny contained 111 genera, including 68 monophyletic genera, 16 non-monophyletic genera, and 27 genera represented by a single species. Among the 84 testable genera (i.e. those with multiple species in the dataset), 80% were recovered as monophyletic (mean bootstrap support: 95%). The PyPHLAWD phylogeny contained 94 genera, including 58 monophyletic genera, 12 non-monophyletic genera, and 24 genera represented by a single species. Among the 70 testable genera, 82% were recovered as monophyletic (mean bootstrap support: 94%). Although both methods produced similar scores for these metrics, we emphasize the SuperCRUNCH phylogeny contained an additional 357 taxa and 17 genera not present in the PyPHLAWD phylogeny.

SuperCRUNCH found all 1,785 UCE loci present for the frog genus *Kaloula* as reported by Alexander et al. (2017). The final supermatrix contained 14 taxa and 22,751 total sequences, and the total run time for this analysis was 1 hr and 35 min (Table 1, Supplemental Table 6). The resulting concatenated phylogeny was congruent with results obtained by Alexander et al. (2017).

For the population-level analyses, we discovered four suitable genes for *Callisaurus* (Cyt-b, CO1, ND2, RAG1) with 284 total sequences representing 20 species and subspecies. The greatest number of terminals occurred in the alignment of Cyt-b (114), followed by RAG1 (71),

ND2 (53), and CO1 (46). We found two suitable mtDNA genes for *Uma* (Cyt-b, CO1) with 210 total sequences representing 10 species and subspecies. The alignment for Cyt-b contained 191 terminals and CO1 contained 19 terminals. Each analysis took less than 3 min of total run time (Table 1, Supplemental Tables 7–8). Concatenation is not straightforward for population-level datasets because different individuals are often sampled for different genes, so all genes were analyzed in RAxML separately. The individual gene trees produced from the six loci were congruent with phylogenetic results presented by Schulte & de Queiroz (2008), Lindell, Méndez-de la Cruz, and Murphy (2005), and Gottscho et al. (2017).

5 | DISCUSSION

SuperCRUNCH is a flexible toolkit for creating, filtering, and manipulating large phylogenetic datasets using data downloaded directly from GenBank/NCBI. We show that SuperCRUNCH generally outperforms the previously best-performing method for supermatrix construction (PyPHLAWD), in terms of building matrices with more sequences and taxa for the same data and same group of organisms. For our main test case (iguanian lizards) the supermatrix from SuperCRUNCH generated a phylogeny showing strong support and considerable congruence with previous trees and taxonomy. In addition to standard supermatrices, SuperCRUNCH is also able to generate datasets for individuals (e.g., for phylogeography and population genetics) and can handle phylogenomic datasets composed of thousands of loci.

Programs that are designed to assemble phylogenetic datasets differ considerably in how starting sequences are obtained. This variation directly impacts the resulting supermatrices. For example, PhyLoTA (Sanderson et al., 2008) and SUPERSMART (Antonelli et al., 2017) use an older GenBank database release (194) that automatically excludes all sequence data made

available since February 2013. From 2015 to 2018, an average of 7.5 million sequences were added to GenBank each year (NCBI, 2019). Given the massive annual increase in sequence data, results from these methods will likely be fundamentally limited. In contrast, PyPHLAWD and SUMAC overcome this issue by using contemporary GenBank releases. However, our results demonstrate that PyPHLAWD, which uses intermediate databases to retrieve sequences, still excludes a considerable amount of sequence data from the final supermatrix (Table 1). SuperCRUNCH has the greatest flexibility in starting material and is currently the only program that can use any set of nucleotide sequence records downloaded directly from NCBI/GenBank. It therefore can circumvent any issues related to older database releases and potentially restrictive taxonomy databases. SuperCRUNCH also offers much finer control over which taxa to include in searches and can help resolve taxonomic discrepancies. These properties may be highly advantageous for groups with rapidly changing taxonomies.

SuperCRUNCH uses sequence-record labels to initially identify loci, which are subsequently validated using stringent orthology filters. The locus-label matching process utilized in SuperCRUNCH requires specifying target loci and search terms, similar to phyloGenerator (Pearse & Purvis, 2013). In contrast, other programs often “identify” loci through the automated clustering of all sequences (PHLAWD, PyPHLAWD, PhyLoTA, PhylotaR, SUPERSMART). This process produces orthologous sequence clusters composed of individual or aggregate loci. This blind clustering method does not allow target loci to be directly specified, which may be problematic under many circumstances. For example, the blind clustering method does not readily identify the sequence clusters, and additional steps such as BLAST searches are required to robustly establish gene identities. Furthermore, clusters falling

below particular sequence thresholds may be discarded, even if they contain the desired target loci.

One clear advantage of the locus-label matching method is that it allows searches to be conducted for each locus without the need to download and validate corresponding sets of reference sequences. Gathering reference sequence sets is tractable for small numbers of loci (<10), but as the scale increases, performing this task for all loci can become laborious (e.g., 68 loci for Iguania) or simply prohibitive (e.g., ~5,000 UCE loci). To guarantee recovery of the target loci for our PyPHLAWD analysis of Iguania, we had to provide validated sequence sets for all loci that were generated using SuperCRUNCH.

However, the locus-label searching method can also be problematic. The limitations of this approach were apparent from our “baited” analysis in PyPHLAWD, which retrieved several orthologous sequences with alternative (but synonymous) gene names for two loci in the Iguania dataset. At the same time, the same PyPHLAWD analysis failed to find 62% of the available mtDNA sequences and also retrieved paralogous sequences for one nuclear locus. These results raise concerns about the effectiveness of the “baited” clustering method. The failure of PyPHLAWD to recover many available mtDNA sequences may be related to a combination of complex sequence records (containing multiple genes) and the BLAST strategy used. Unless specified (using the -task flag), the default algorithm used by nucleotide BLAST is megablast, which is best for finding highly similar sequences in intraspecific searches. An important feature of SuperCRUNCH is the ability to specify the BLAST algorithm. For all our searches we used discontiguous megablast, which performs substantially better for interspecific searches (Ma, Tromp, & Li 2002; Camacho et al. 2009). Our SuperCRUNCH analysis of the directly downloaded Iguania sequences included a mix of whole mitochondrial genomes and complex

mtDNA records, and we successfully recovered the intended targets from these records. Clearly, there are pros and cons to each locus-search method. Nevertheless, the SuperCRUNCH strategy of locus-label matching followed by orthology filtering offers considerable flexibility for targeted locus searching, and performed as well or better than baited clustering for obtaining sequences (Table 1, Supplemental Tables 1, 2).

Multiple sequence alignment is perhaps the most important and overlooked step in supermatrix construction, as poorly aligned sequences can lead to inaccurate tree topologies (Ogden & Rosenberg 2006). Many alternative programs only offer alignments using MAFFT (PyPHLAWD, SUMAC, SUPERSMART). On the positive side, MAFFT is fast, scalable, and generally reliable (Katoh et al., 2002; Katoh & Standley, 2013). In SuperCRUNCH, we implement MAFFT along with several other aligners (CLUSTAL-O, MACSE, MUSCLE). These alternative aligners may perform better for different types of loci and at different taxonomic scales. For example, in the Iguania dataset we observed that MAFFT failed to accurately align shorter sequences present in both 12S and 16S, whereas CLUSTAL-O placed these fragments correctly. One important alignment feature is the inclusion of MACSE (Ranwez et al., 2018), which aligns coding sequences while allowing for multiple frameshifts or stop codons. Translation aligners generally require all sequences be provided in the first forward frame and with complete final codons, an onerous task for GenBank sequence data. SuperCRUNCH automates this editing process and also produces inputs for the paired-translation alignment feature available in MACSE. Given that SuperCRUNCH simplifies the use of the available aligners, we encourage comparisons of aligner performance to ensure high quality alignments are being produced (see for example Thompson, Linard, Lecompte, & Poch 2011).

Another important and more general issue is the accuracy of the sequence data in GenBank. For example, errors may arise through incorrect uploading of data, misidentified specimens, contamination, and other lab errors. With regards to data errors, we know of cases in which taxon labels in GenBank were inadvertently switched during uploading, such that the sequence data for a gene in two species in two genera were transposed, and thus were incorrect and potentially misleading. With regards to contamination, we identified and removed a human 12S mtDNA sequence labeled as a lizard in our main iguanian analysis (HM040901.1: *Laudakia tuberculata*). The contamination filter in SuperCRUNCH can detect and eliminate some problems of this kind, but it cannot readily identify cases of misidentified or mislabeled sequences within the focal group. Misidentified specimens are perhaps the most difficult problem to detect, particularly when the incorrect identity occurs within a shallow taxonomic scale (e.g., a specimen assigned to the wrong species within the same genus or family). Although orthology filtering can be used to correctly establish gene identities, parallel approaches for identifying inaccurate taxon labeling within the focal group are currently lacking. One potential solution is to simply check the resulting trees for incongruence with previous hypotheses, especially the studies that were the major sources for the sequences used in a given taxon. Our program MonoPhylo may be helpful in testing congruence with previous taxonomies and hypotheses and pinpointing problematic taxa. Overall, we think that data accuracy is a general problem for the supermatrix approach regardless of the methods used. Inaccurate sequence records remain a challenging problem that would be a useful focus for future studies involving supermatrix construction.

SuperCRUNCH analyses are modular, highly reproducible, and output key information after each step, offering many entry points. As such, SuperCRUNCH can be incorporated into

bioinformatics pipelines and used in conjunction with features of other currently available programs. For example, SuperCRUNCH can generate alignments for thousands of loci using multiple aligner options, and therefore can supply the alignment step for any phylogenetic data processing pipeline. We also demonstrated that SuperCRUNCH can be used to create validated reference sequence sets which are compatible with programs such as PyPHLAWD. We appreciate that other programs offer important features that may serve different needs (e.g., SUPERSMART performs phylogenetic analyses on the supermatrices that it generates). We emphasize these methods should be viewed as complementary resources. Given the rapidly growing volume of sequence data available on GenBank (NCBI, 2019), there is a need to continue developing improved bioinformatics approaches to mine and manage phylogenetic datasets.

ACKNOWLEDGMENTS

For financial support, we thank U.S. National Science Foundation Grant DEB 1655690.

AUTHORS' CONTRIBUTIONS

DMP and JJW conceived the ideas; DMP designed the methodology, wrote all code, and analyzed the data; DMP and JJW wrote the manuscript.

DATA ACCESSIBILITY

The complete set of materials and instructions required to replicate our analyses, including all input and output files from each step, is freely available from the Open Science Framework

(https://osf.io/bpt94/). This material includes all analyses run using SuperCRUNCH and PyPHLAWD (with associated wiki tutorials), final alignments and supermatrices, and resulting phylogenies. SuperCRUNCH is open-source and freely available at https://github.com/dportik/SuperCRUNCH. MonoPhylo is open-source and freely available at https://github.com/dportik/MonoPhylo.

REFERENCES

Alexander, A. A., Su, Y.-C., Oliveros, C. H., Olson, K. V., Travers, S. L., & Brown, R. M. (2016). Genomic data reveals potential for hybridization, introgression, and incomplete lineage sorting to confound phylogenetic relationships in an adaptive radiation of narrow-mouth frogs. *Evolution, 71,* 475–488.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology, 215,* 403– 410.

AmphibiaWeb. 2019. Electronic Database accessible at https://amphibiaweb.org. University of California, Berkeley, CA, USA.

Antonelli, A., Hettling, H., Condamine, F. L., Vos, K., Nilsson, R. H., Sanderson, M. J., Sauquet, H., Scharn, R., Silvestro, D., Töpel, M., Bacon, C. D., Oxelman, B., & Vos, R. A. (2017). Toward a self-updating platform for estimating rates of speciation and migration, ages, and relationships of taxa. *Systematic Biology*, *66,* 152–166.

Bennett, D. J., Hettling, H., Silvestro, D., Zizka, A., Bacon, C. D., Faurby, S., Vos, R. A., & Antonelli, A. (2018). phylotaR: an automated pipeline for retrieving orthologous DNA sequences from GenBank in R. *Life, 8,* 20.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., & Madden, T. L. (2009). BLAST+: architecture and applications. *BMC Bioinformatics*, *10*, 421.

Capella-Gutiérrez, S., Silla-Martínez, J. M., & Gabaldón, T. (2009). trimAl: a tool for automated trimming in large-scale phylogenetic analyses. *Bioinformatics, 25,* 1972–1973.

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, *32*, 1792–1797.

Freyman, W. A. (2015). SUMAC: Constructing phylogenetic supermatrices and assessing partially decisive taxon coverage. *Evolutionary Bioinformatics*, *11*, 263–266.

Frost, D. R. (2019). Amphibian Species of the World: an Online Reference. Version 6.0. Electronic Database accessible at http://research.amnh.org/herpetology/amphibia/index.html. American Museum of Natural History, New York, USA.

Gottscho, A. D., Wood, D. A., Vandergast, A. G., Lemos-Espinal, J., Gatesy, J., and Reeder, T. W. (2017). Lineage diversification of fringe-toed lizards (Phrynosomatidae: *Uma notata* complex) in the Colorado Desert: delimiting species in the presence of gene flow. *Molecular Phylogenetics and Evolution*, *106*, 103–117.

Huerta-Cepas, J., Serra, F., & Bork, P. (2016). ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Molecular Biology and Evolution*, *33*, 1635–1638.

Katoh, K., Misawa, K., Kuma, K., & Miyata, T. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, *30*, 3059–3066.

Katoh, K., & Standley, D. M. (2013). MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular Biology and Evolution*, *30*, 772–780.

Li, W., & Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, *22*, 1658–1659.

Lindell, J., Méndez-de la Cruz, F. R., & Murphy, R. W. (2005). Deep genealogical history without population differentiation: discordance between mtDNA and allozyme divergence in the zebra-tailed lizard (*Callisaurus draconoides*). *Molecular Phylogenetics and Evolution*, *36*, 682–694.

Ma, B., Tromp, J., & Li, M. (2002). PatternHunter: faster and more sensitive homology search. *Bioinformatics, 18*, 440–445.

NCBI. (2019). GenBank and WGS Statistics. Database accessible at https://www.ncbi.nlm.nih.gov/genbank/statistics/. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information.

Ogden, T. H., & Rosenberg, M. S. (2006). Multiple sequence alignment accuracy and phylogenetic inference. *Systematic Biology, 55*, 314–328.

Pearse, W. D., & Purvis, A. (2013). phyloGenerator: An automated phylogeny generation tool for ecologists. *Methods in Ecology and Evolution, 4*, 692–698.

Portik, D. M., Wood Jr., P. L., Grismer, J. L., Stanley, E. L., & Jackman, T. R. (2012). Identification of 104 rapidly-evolving nuclear protein-coding markers for amplification across scaled reptiles using genomic resources. *Conservation Genetics Resources, 4*, 1−10.

Pyron, R. A., Burbrink, F. T., & Wiens, J. J. (2013). A phylogeny and revised classification of Squamata, including 4161 species of lizards and snakes. *BMC Evolutionary Biology, 13*, 93.

de Queiroz, A., & Gatesy, J. (2007). The supermatrix approach to systematics. *Trends in Ecology and Evolution, 22*, 34–41.

Ranwez, V., Douzery, E. J. P., Cambon, C., Chantret, N., & Delsuc, F. (2018). MACSE v2: toolkit for the alignment of coding sequences accounting for frameshifts and stop codons. *Molecular Biology and Evolution, 35*, 2582–2584.

Sanderson, M. J., Boss, D., Chen, D., Cranston, K. A., & Wehe, A. (2008). The PhyLoTA browser: processing GenBank for molecular phylogenetics research. *Systematic Biology, 57*, 335–346.

Schulte II, J. A., & de Queiroz, K. (2008). Phylogenetic relationships and heterogeneous evolutionary processes among phrynosomatine sand lizards (Squamata, Iguanidae) revisited. *Molecular Phylogenetics and Evolution, 47,* 700–716.

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., Thompson, J. D., & Higgins, D. G. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, *7*, 539.

Smith, S. A., Beaulieu, J. M., & Donoghue, M. J. (2009). Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. *BMC Evolutionary Biology*, *9*, 37.

Smith, S. A., & Walker, J. F. (2018). PyPHLAWD: A python tool for phylogenetic dataset construction. *Methods in Ecology and Evolution*, *Early View*.

Stamatakis, A. (2014). RAxML version 8: a toolf for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics, 30,* 1312–1313.

Stelzer, G., Rosen, N., Plaschkes, I., Zimmerman, S., Twik, M., Fishilevich, S., Stein, T.I., Nudel, R., Lieder, I., Mazor, Y., Kaplan, S., Dahary, D., Warshawsky, D., Guan-Golan, Y., Kohn, A., Rappaport, N., Safran, M., & Lancet, D. 2016. The GeneCards suite: from gene data mining to disease genome sequence analyses. *Current Protocols in Bioinformatics*, *54,* 1.30.1–1.30.33.

Streicher, J. W., Schulte, J. A., & Wiens, J. J. (2016). How should genes and taxa be sampled for phylogenomic analyses with missing data? An empirical study in iguanian lizards. *Systematic Biology, 65,* 128–145.

Thompson, J. D., Linard, B., Lecompte, O., & Poch, O. (2011). A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS ONE, 6*, e18093.

Townsend, T. M., Alegre, R. E., Kelley, S. T., Wiens, J. J., & Reeder, T. W. (2008). Rapid development of multiple nuclear loci for phylogenetic analysis using genomic resources: an example from squamate reptiles. *Molecular Phylogenetics and Evolution, 47,* 129–142.

Uetz, P., Freed, P., & Hošek, J. (2018). The Reptile Database. Available at: http://www.reptile-database.org.

Zheng, Y., & Wiens, J. J. (2016). Combining phylogenomic and supermatrix approaches, and a time-calibrated phylogeny for squamate reptiles (lizards and snakes) based on 52 genes and 4,162 species. *Molecular Phylogenetics and Evolution, 94,* 537–547.

FIGURE CAPTIONS

FIGURE 1 Taxon filtering and locus extraction workflow of SuperCRUNCH.

FIGURE 2 Orthology filtering workflow of SuperCRUNCH.

FIGURE 3 Sequence quality and selection workflow of SuperCRUNCH.

FIGURE 4 Alignment workflow of SuperCRUNCH.

FIGURE 5 Post-alignment workflow of SuperCRUNCH.

TABLE 1 Summary of characteristics for supermatrices obtained using SuperCRUNCH and PyPHLAWD for each group.

Group	Method	Objective	Starting Sequence Set	Taxa	Loci Found	Total Sequences	CPU Time (H:M:S)
Iguania	SuperCRUNCH	Supermatrix	NCBI direct download	1,426	66	13,419	10:13:36
	SuperCRUNCH	Supermatrix	PyPHLAWD database	1,358	66	12,654	04:29:02
	PyPHLAWD	Supermatrix	PyPHLAWD database	1,069	65	10,397	00:18:15
Dipsacales	SuperCRUNCH	Supermatrix	PyPHLAWD database	651	4	1,598	00:37:51
	PyPHLAWD	Supermatrix	PyPHLAWD database	641	4	1,510	00:01:16
Kaloula	SuperCRUNCH	UCE Supermatrix	NCBI direct download	14	1,785	22,751	01:35:16
Callisaurus	SuperCRUNCH	Population-level	NCBI direct download	20	4	284	00:02:27
Uma	SuperCRUNCH	Population-level	NCBI direct download	10	2	210	00:01:52

Supplemental Table 1. Sequences retained for each locus after each major step in SuperCRUNCH using iguanian lizard data downloaded directly from NCBI, and the final sequence counts from PyPHLAWD and SuperCRUNCH analyses using the smaller PyPHLAWD-generated Iguania sequence set. Bold indicates loci for which PyPHLAWD recovered more sequences than SuperCRUNCH, which resulted from gene name synonomy (R35, ZEB2) or the inclusion of paralogous sequences (ENC1).

Location	Locus	Locus filtering	Orthology filtering	Length filtering	Final sequence selection	PyPHLAWD	SuperCRUNCH
mtDNA	12S	2,735	2,601	2,598	530	325	478
	16S	4,015	3,980	3,800	578	9	541
	CO1	3,945	2,731	2,543	484	-	342
	CYTB	9,464	9,464	9,463	513	348	480
	ND1	2,659	2,537	1,897	202	109	182
	ND2	11,381	11,380	11,377	1,005	523	923
	ND4	4,675	4,648	4,648	551	1	494
Nuclear	ADNP	169	169	169	145	142	142
	AHR	160	160	160	137	133	133
	AKAP9	223	223	223	184	177	180
	AMEL	49	49	49	14	10	13
	BACH1	551	551	551	187	183	183
	BACH2	31	31	31	31	29	30
	BDNF	1,269	1,269	1,269	394	375	379
	BHLHB2	171	171	171	149	146	146
	BMP2	168	168	168	143	140	140
	CAND1	173	173	173	149	145	145
	CARD4	162	162	162	141	139	139
	CILP	167	167	167	145	143	143
	CMOS	1,312	1,312	1,312	467	443	443
	CXCR4	166	166	166	145	143	143
	DLL1	163	163	163	140	140	140
	DNAH3	640	639	639	141	137	137
	ECEL1	221	221	221	150	150	150
	ENC1	45	45	45	45	**94**	41
	EXPH5	651	651	651	149	148	148
	FSHR	170	170	170	148	145	145

FSTL5	169	169	169	144	141	141
GALR1	157	157	157	138	137	137
GHSR	152	152	152	132	132	132
GPR37	166	166	166	144	140	140
HLCS	162	162	162	139	137	137
INHIBA	166	166	166	145	140	140
KIAA1217	0	-	-	-	-	-
KIAA1549	0	-	-	-	-	-
KIAA2018	291	291	291	15	15	15
KIF24	454	441	441	139	139	139
LRRN1	154	154	154	132	132	132
LZTS1	140	140	140	122	122	122
MC1R	657	657	657	40	40	40
MKL1	181	181	181	153	114	114
MLL3	145	145	145	123	123	123
MSH6	191	191	191	168	163	164
MXRA5	369	369	369	107	104	104
MYH2	0	-	-	-	-	-
NGFB	498	498	498	173	169	169
NKTR	760	757	757	227	215	221
NOS1	219	219	219	9	9	9
NT3	886	886	886	368	350	355
PDC	203	203	203	19	19	19
PNN	706	706	706	277	269	269
PRLR	1,145	1,142	1,140	440	420	420
PTGER4	270	270	270	149	147	147
PTPN	202	202	202	115	114	114
R35	1,011	1,010	1,010	303	**300**	288
RAG1p1	3,139	2,555	2,555	455	344	429
RAG1p2	3,139	1,067	894	441	290	407
RAG2	2	-	-	-	-	-
REV3L	30	30	30	30	28	29
RHO	377	377	377	58	38	38
SLC30A1	170	170	170	147	144	144
SLC8A1	169	169	169	148	144	144
SLC8A3	168	168	168	146	143	143
SNCAIP	700	700	700	252	249	249
SOCS5	78	78	78	18	17	17
TRAF6	241	241	241	152	149	149
UBN1	259	259	259	151	147	148

VCPIP	148	148	148	133	131	131
ZEB2	269	269	269	157	**164**	154
ZFP36L1	166	166	166	143	141	141

Supplemental Table 2. The number of total sequences recovered for each locus from the PyPHLAWD baited analysis and a SuperCRUNCH analysis of Dipsacales based on the PyPHLAWD-generated starting sequence set.

Locus	PyPHLAWD	SuperCRUNCH
ITS	556	584
matK	322	352
rbcl	299	299
trnL-trnF	333	363

Supplemental Table 3. Summary of elapsed times to run major steps in SuperCRUNCH for the Iguania sequence dataset downloaded directly from NCBI.

Module	Step information	Elapsed time (H:M:S)
Taxa_Assessment.py	Search of 8,785,378 records	0:22:41
Rename_Merge.py	Using -m flag	0:04:32
Parse_Loci.py	Searched 160,795 records for 69 loci	0:08:42
Cluster_Blast_Merge.py	Performed on 57 nuclear loci	0:28:18
Blast_Extract.py	Performed on 7 mtDNA loci, 2 nuclear loci	0:55:14
Contamination_Filter.py	Performed on 7 mtDNA loci	0:00:07
Filter_Seqs_and_Species.py	-t standard; 57 nuclear loci	0:01:15
	-t vmtdna; 5 mtDNA loci	0:05:26
	-t noncoding; 2 mtDNA loci	0:01:01
Make_Acc_Table.py	Performed on 58 nuclear loci, 7 mtDNA loci	0:00:01
Adjust_Direction.py	Performed on 58 nuclear loci, 7 mtDNA loci	0:58:51
Align.py	-a all; 63 coding loci	1:48:57
	-a clustalo --accurate; 2 noncoding loci	0:53:01
	-a macse; 49 nuclear loci	1:43:10
	-a macse --pass_fail; 9 nuclear loci	0:14:16
	-a macse; 1 mtDNA locus	1:31:03
	-a macse --pass_fail; 4 mtDNA loci	0:56:39
Coding_Translation_Tests.py	-t standard; 57 nuclear loci	0:00:12
	-t vmtdna; 5 mtDNA loci	0:00:03
Relabel_Fasta.py	Performed on 58 nuclear loci, 7 mtDNA loci	0:00:01
Trim_Alignments.py	Performed on 58 nuclear loci, 7 mtDNA loci	0:00:05
Concatenation.py	Performed on 58 nuclear loci, 7 mtDNA loci	0:00:01
Total CPU time	**Includes all steps above**	**10:13:36**

Supplemental Table 4. Summary of elapsed times to run major steps in SuperCRUNCH for the Iguania dataset obtained from PyPHLAWD.

Module	Step information	Elapsed time (H:M:S)
Taxa_Assessment.py	--no_subspecies; Search of 134,028 records	0:00:18
Rename_Merge.py	Used -m flag	0:00:05
Parse_Loci.py	--no_subspecies; Searched 133,962 records for 69 loci	0:01:59
Cluster_Blast_Merge.py	Performed on 58 nuclear loci	0:22:44
Blast_Extract.py	Performed on 7 mtDNA loci, 2 nuclear loci	0:48:36
Filter_Seqs_and_Species.py	-t length; 67 loci	0:24:11
Make_Acc_Table.py	Performed on 67 loci	0:00:01
Adjust_Direction.py	Performed on 67 loci	1:18:30
Align.py	-a all; 67 loci	1:36:36
Relabel_Fasta.py	Performed on 67 loci	0:00:01
Concatenation.py	Performed on 67 loci	0:00:01
Total CPU time	**Includes all steps above**	**4:29:02**

Supplemental Table 5. Summary of elapsed times to run major steps in SuperCRUNCH for the Dispsacales sequence set obtained from PyPHLAWD.

Module	Step information	Elapsed time (H:M:S)
Taxa_Assessment.py	Searched 12,348 records	0:00:01
Parse_Loci.py	Searched 12,348 records for 4 loci	0:00:01
Blast_Extract.py	Performed on 4 loci	0:00:05
Filter_Seqs_and_Species.py	-t length; 4 loci	0:01:57
Make_Acc_Table.py	Performed on 4 loci	0:00:01
Adjust_Direction.py	Performed on 4 loci	0:16:05
Align.py	-a all; 4 loci	0:19:39
Relabel_Fasta.py	-r species; Performed on 4 loci	0:00:01
Concatenation.py	Concatenated 4 loci	0:00:01
Total CPU time	**Includes all steps above**	**0:37:51**

Supplemental Table 6. Summary of elapsed times to run major steps in SuperCRUNCH for the UCE sequence set obtained for the frog genus *Kaloula*.

Module	Step information	Elapsed time (H:M:S)
Taxa_Assessment.py	Searched 38,568 records	0:00:05
Parse_Loci.py	Searched 28,837 records for 5,041 loci	0:26:19
Cluster_Blast_Extract.py	Performed on 1,785 loci	0:05:54
Filter_Seqs_and_Species.py	Length method; 1,785 loci	0:00:31
Make_Acc_Table.py	Performed on 1,785 loci, 14 taxa	0:00:01
Adjust_Direction.py	Performed on 1,785 loci	0:06:46
Align.py	-a mafft; 4 loci	0:18:16
Relabel_Fasta.py	-r species; Performed on 1,785 loci	0:00:02
Trim_Alignments.py	Performed on 1,785 loci	0:00:19
Concatenation.py	Concatenated 1,785 loci	0:00:03
Total CPU time	**Includes all steps above**	**1:35:16**

Supplemental Table 7. Summary of elapsed times to run major steps in SuperCRUNCH for the *Callisaurus* population-level data set.

Module	Step information	Elapsed time (H:M:S)
Fasta_Get_Taxa.py	Searched 82,557 records	0:00:01
Taxa_Assessment.py	Searched 82,557 records	0:00:02
Parse_Loci.py	Searched 2,882 records for 69 loci	0:00:02
Reference_Blast_Extract.py	Performed on 6 loci	0:00:08
Filter_Seqs_and_Species.py	Length method; 4 loci	0:00:01
Adjust_Direction.py	Performed on 4 loci	0:00:01
Align.py	-a all; 4 loci	0:02:10
Relabel_Fasta.py	-r species_acc; 4 loci	0:00:01
Fasta_Convert.py	Performed on 4 loci	0:00:01
Total CPU time	**Includes all steps above**	**0:02:27**

Supplemental Table 8. Summary of elapsed times to run major steps in SuperCRUNCH for the *Uma* population-level data set.

Module	Step information	Elapsed time (H:M:S)
Fasta_Get_Taxa.py	Searched 82,557 records	0:00:01
Taxa_Assessment.py	Searched 82,557 records	0:00:06
Parse_Loci.py	Searched 1,259 records for 69 loci	0:00:01
Reference_Blast_Extract.py	Performed on 2 loci	0:00:12
Filter_Seqs_and_Species.py	Translate method; 2 loci	0:00:01
Adjust_Direction.py	Performed on 2 loci	0:00:02
Align.py	-a all; 2 loci	0:01:27
Relabel_Fasta.py	-r species_acc; Performed on 2 loci	0:00:01
Fasta_Convert.py	Performed on 2 loci	0:00:01
Total CPU time	**Includes all steps above**	**0:01:52**

Supplemental Table 9. Summary of the content, monophyly status, and associated bootstrap support for iguanian genera obtained from the phylogenies resulting from SuperCRUNCH or PyPHLAWD supermatrices.

	SuperCRUNCH			PyPHLAWD		
Genus	Taxa	Monophyletic	Support	Taxa	Monophyletic	Support
Acanthocercus	5	N	-	5	N	-
Acanthosaura	4	Y	100	3	Y	93
Agama	38	N	-	32	Y	100
Amblyrhynchus	1	-	-	1	-	-
Amphibolurus	2	Y	100	2	Y	100
Anisolepis	2	Y	100	1	-	-
Anolis	324	Y	95	208	Y	96
Aphaniotis	1	-	-	-	-	-
Archaius	1	-	-	1	-	-
Basiliscus	4	Y	96	4	Y	75
Brachylophus	3	Y	100	2	Y	100
Bradypodion	17	Y	100	17	Y	97
Bronchocela	5	Y	100	1	-	-
Brookesia	28	Y	100	24	Y	98
Bufoniceps	1	-	-	-	-	-
Cachryx	2	Y	100	2	Y	100
Callisaurus	1	-	-	1	-	-
Calotes	15	Y	93	9	Y	100
Calumma	31	Y	45	29	Y	36
Ceratophora	5	Y	85	-	-	-
Chalarodon	1	-	-	1	-	-
Chamaeleo	14	Y	100	13	Y	99
Chelosania	1	-	-	1	-	-
Chlamydosaurus	1	-	-	1	-	-
Conolophus	3	Y	100	3	Y	100
Cophosaurus	1	-	-	1	-	-
Cophotis	2	Y	100	-	-	-
Coryphophylax	1	-	-	-	-	-
Corytophanes	3	Y	100	3	Y	100
Crotaphytus	9	Y	100	9	Y	100
Ctenoblepharys	1	-	-	-	-	-
Ctenophorus	24	N	-	19	Y	100
Ctenosaura	13	Y	100	13	Y	100

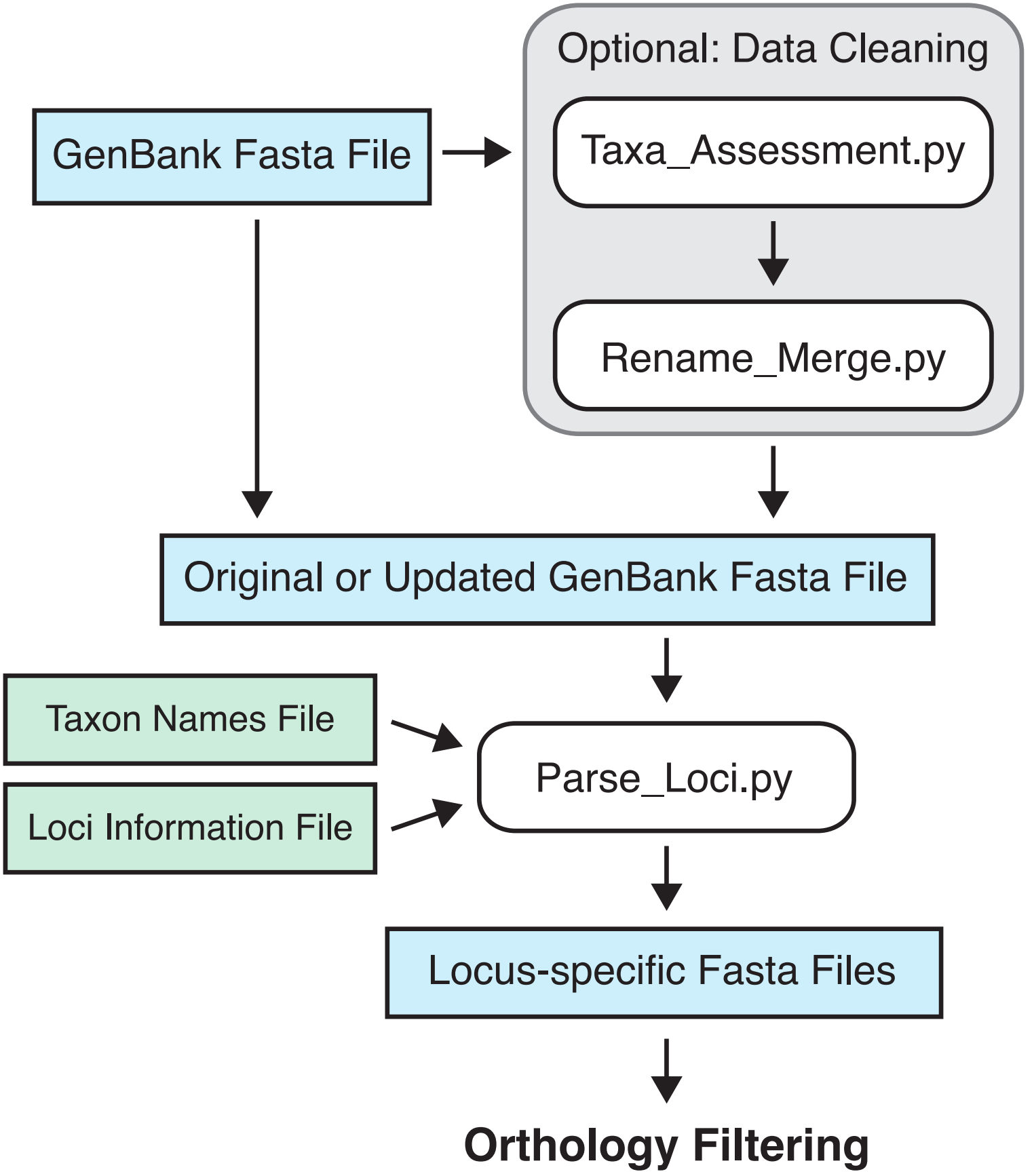
Cyclura	8	Y	100	5	Y	100
Diplolaemus	4	Y	100	4	Y	94
Diporiphora	16	Y	48	14	Y	87
Dipsosaurus	1	-	-	1	-	-
Draco	34	Y	99	3	N	-
Enyalioides	14	N	-	14	N	-
Enyalius	9	N	-	9	Y	76
Eurolophosaurus	3	Y	97	1	-	-
Furcifer	21	Y	99	19	Y	91
Gambelia	3	Y	100	3	Y	100
Gonocephalus	9	Y	84	9	Y	93
Gowidon	2	Y	66	-	-	-
Holbrookia	5	Y	100	4	Y	93
Hoplocercus	1	-	-	1	-	-
Hydrosaurus	3	Y	100	2	Y	100
Hypsilurus	5	N	-	3	Y	99
Iguana	2	Y	100	2	Y	74
Intellagama	1	-	-	-	-	-
Japalura	11	N	-	6	N	-
Kinyongia	21	N	-	20	Y	99
Laemanctus	1	-	-	1	-	-
Laudakia	6	Y	100	1	-	-
Leiocephalus	15	Y	96	12	Y	100
Leiolepis	5	Y	100	4	Y	100
Leiosaurus	4	Y	100	4	Y	100
Liolaemus	181	Y	97	151	Y	98
Lophocalotes	2	Y	100	-	-	-
Lophognathus	2	Y	85	2	Y	94
Lophosaurus	3	Y	39	-	-	-
Lyriocephalus	1	-	-	-	-	-
Malayodracon	1	-	-	1	-	-
Mantheyus	1	-	-	-	-	-
Microlophus	21	Y	99	20	Y	100
Moloch	1	-	-	1	-	-
Morunasaurus	1	-	-	1	-	-
Nadzikambia	2	Y	100	2	N	-
Oplurus	6	N	-	6	N	-
Otocryptis	2	N	-	1	-	-
Palleon	2	Y	93	-	-	-
Paralaudakia	6	Y	100	3	Y	99

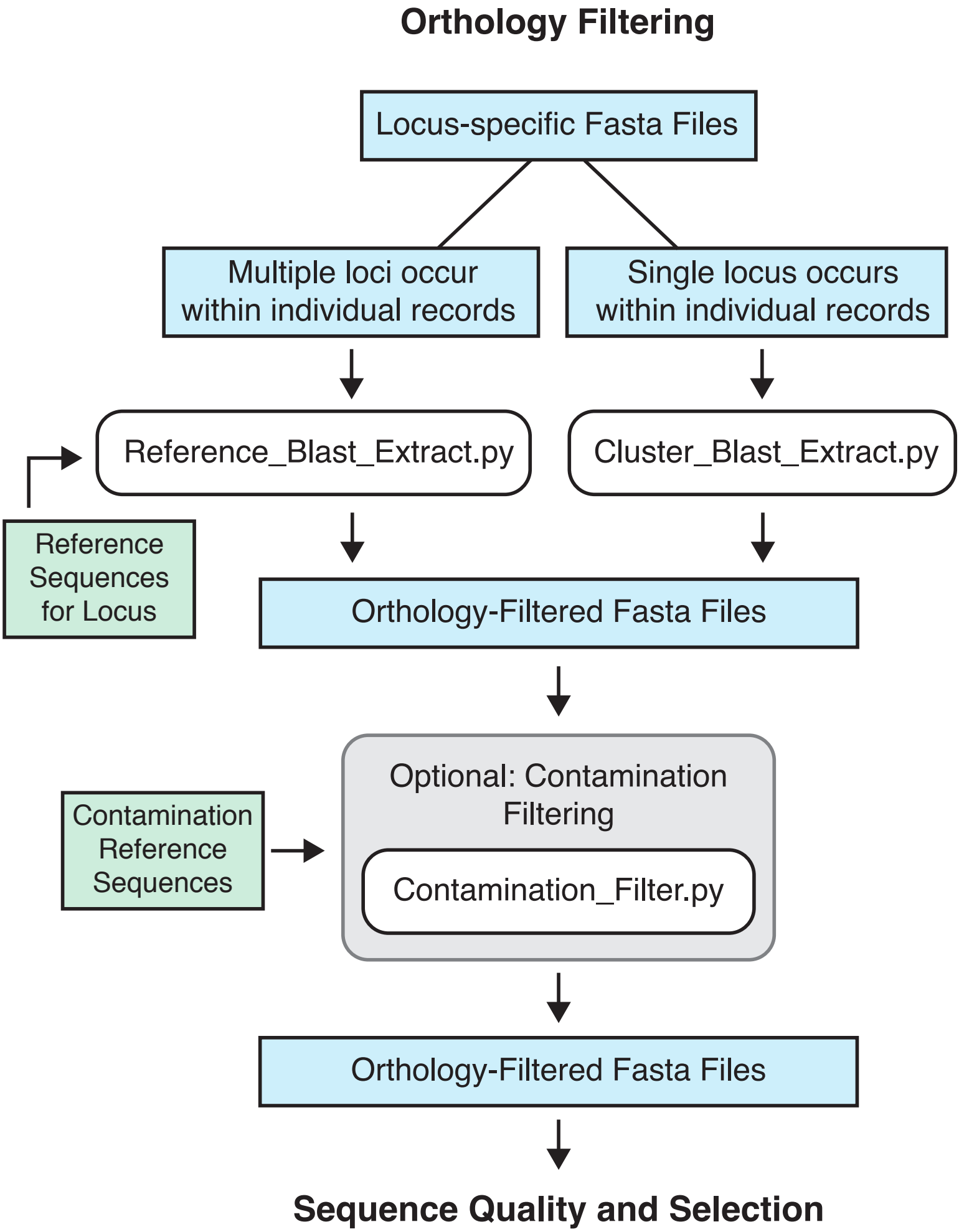
Petrosaurus	3	Y	100	3	Y	100
Phoxophrys	1	-	-	-	-	-
Phrynocephalus	27	Y	100	20	Y	100
Phrynosoma	18	Y	100	18	Y	100
Phymaturus	44	Y	100	34	Y	100
Physignathus	1	-	-	1	-	-
Plica	3	Y	95	3	N	-
Pogona	7	Y	100	6	Y	100
Polychrus	8	Y	99	8	Y	100
Pristidactylus	6	N	-	6	N	-
Psammophilus	1	-	-	-	-	-
Pseudocalotes	15	N	-	2	N	-
Pseudotrapelus	6	Y	100	6	Y	100
Ptyctolaemus	2	Y	100	-	-	-
Rankinia	1	-	-	1	-	-
Rhampholeon	15	Y	95	12	N	-
Rieppeleon	3	Y	100	3	Y	58
Saara	3	Y	100	2	Y	100
Salea	2	N	-	-	-	-
Sarada	3	Y	100	3	Y	100
Sauromalus	4	Y	100	4	Y	45
Sceloporus	89	Y	92	86	Y	100
Sitana	5	Y	100	5	Y	100
Stellagama	1	-	-	1	-	-
Stenocercus	40	Y	100	10	Y	100
Strobilurus	1	-	-	1	-	-
Trapelus	8	Y	100	4	Y	98
Trioceros	33	Y	100	26	Y	99
Tropidurus	28	N	-	24	N	-
Tympanocryptis	9	Y	100	7	Y	100
Uma	6	Y	100	6	Y	100
Uracentron	2	N	-	1	-	-
Uranoscodon	1	-	-	1	-	-
Uromastyx	14	Y	100	13	Y	100
Urosaurus	8	Y	100	8	Y	100
Urostrophus	2	N	-	2	N	-
Uta	3	Y	100	3	Y	100
Xenagama	3	Y	97	3	Y	100

Supplemental Table 10. Summary of the content, monophyly status, and associated bootstrap support for iguanian families and subfamilies obtained from the phylogenies resulting from SuperCRUNCH or PyPHLAWD supermatrices.

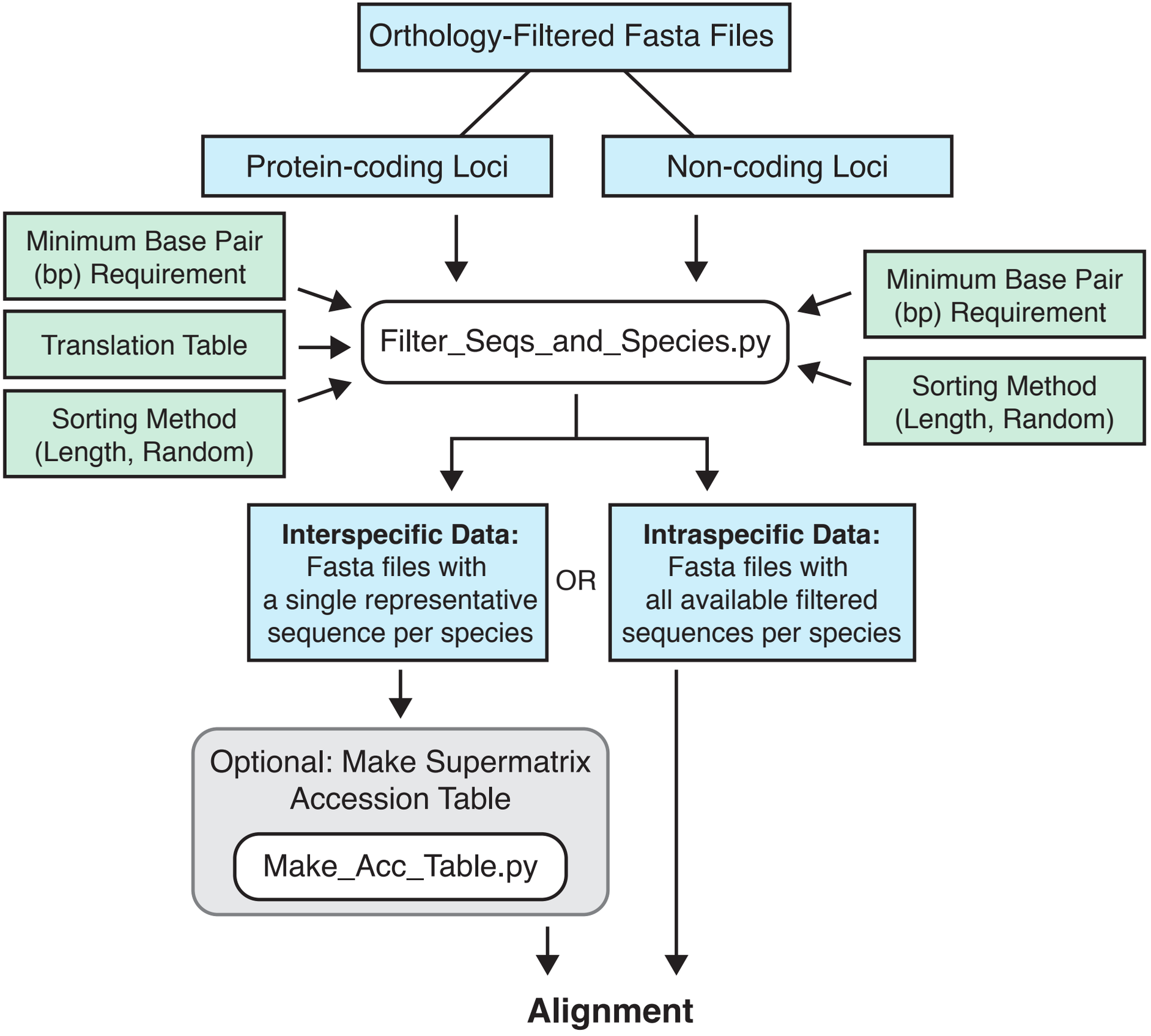
		SuperCRUNCH			PyPHLAWD		
Rank	Classification	Taxa	Monophyletic	Support	Taxa	Monophyletic	Support
Family	Agamidae	325	Y	89	197	N	-
	Chamaeleonidae	188	Y	100	166	Y	94
	Corytophanidae	8	Y	100	8	Y	100
	Crotaphytidae	12	Y	100	12	Y	100
	Dactyloidae	324	Y	95	208	Y	96
	Hoplocercidae	16	Y	100	16	Y	100
	Iguanidae	37	Y	83	33	Y	100
	Leiocephalidae	15	Y	96	12	Y	100
	Leiosauridae	27	Y	100	26	Y	100
	Liolaemidae	226	Y	77	185	Y	97
	Opluridae	7	Y	100	7	Y	100
	Phrynosomatidae	134	Y	100	130	Y	100
	Polychrotidae	8	Y	99	8	Y	100
	Tropiduridae	99	Y	98	61	Y	98
Subfamily	Agaminae	101	Y	100	75	Y	99
	Amphibolurinae	76	Y	100	58	Y	100
	Draconinae	123	Y	95	43	Y	99
	Enyaliinae	13	Y	96	12	Y	96
	Hydrosaurinae	3	Y	100	2	Y	100
	Leiolepidinae	5	Y	100	4	Y	100
	Leiosaurinae	14	Y	100	14	Y	98
	Phrynosomatinae	31	Y	100	30	Y	100
	Sceloporinae	103	Y	100	100	Y	100
	Uromastycinae	17	Y	100	15	Y	100

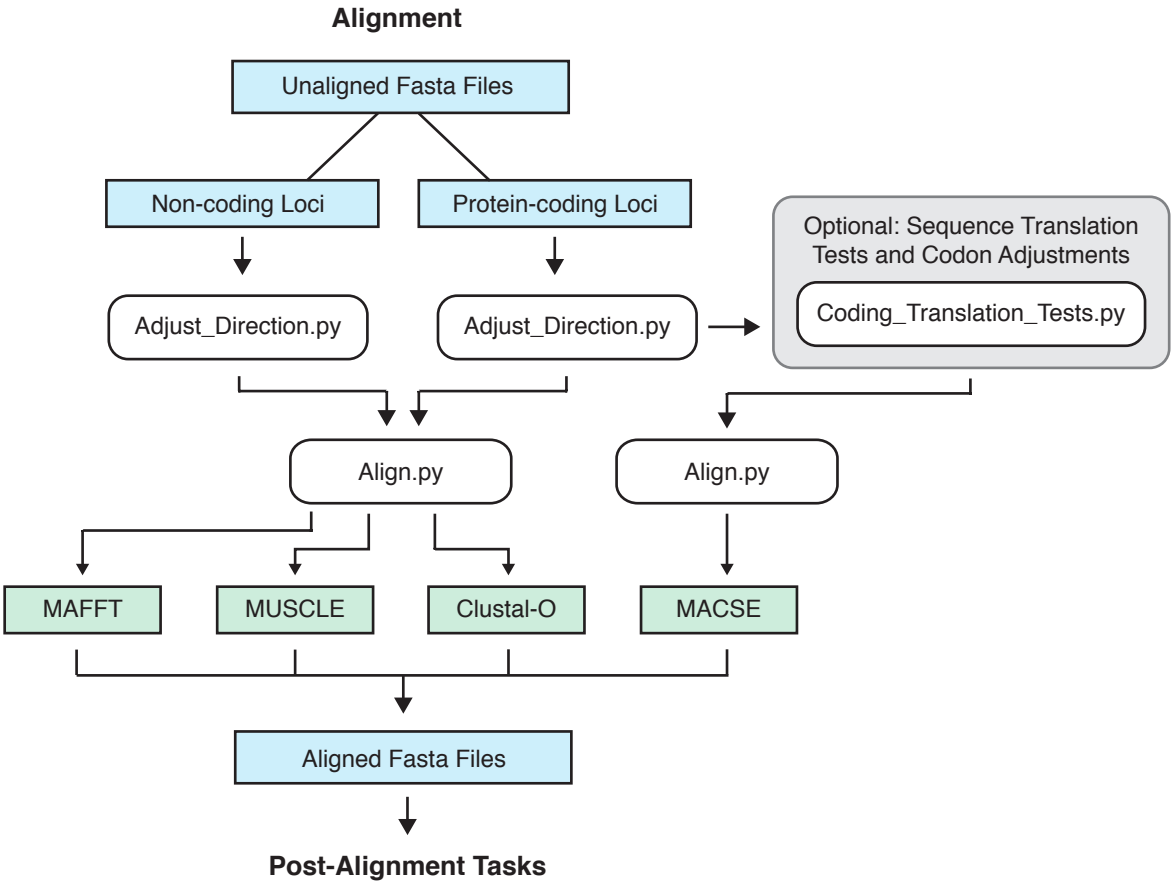
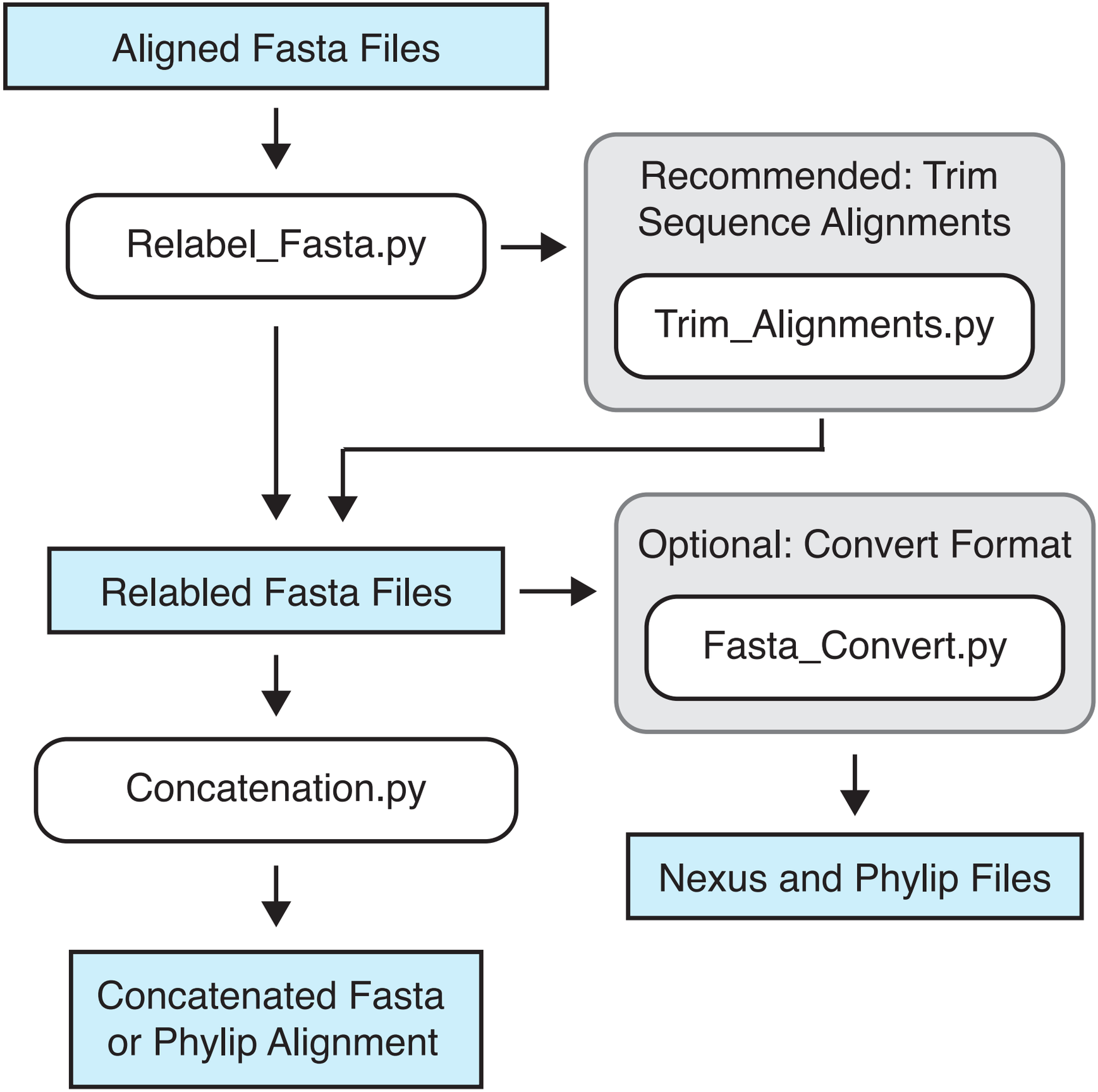
Taxon Filtering and Locus Extraction




Orthology Filtering
Locus-specific Fasta Files
Multiple loci occur within individual records
Single locus occurs within individual records
Reference_Blast_Extract.py
Cluster_Blast_Extract.py
Reference Sequences for Locus
Orthology-Filtered Fasta Files
Optional: Contamination Filtering
Contamination Reference Sequences
Contamination_Filter.py
Orthology-Filtered Fasta Files
Sequence Quality and Selection

Sequence Quality and Selection




Alignment
Unaligned Fasta Files
Non-coding Loci
Protein-coding Loci
Optional: Sequence Translation Tests and Codon Adjustments
Adjust_Direction.py
Adjust_Direction.py
Coding_Translation_Tests.py
Align.py
Align.py
MAFFT
MUSCLE
Clustal-O
MACSE
Aligned Fasta Files
Post-Alignment Tasks

Post-Alignment Tasks