SOPHI
Structurally Optimized PHylo-Informatics.

contact address dc0357548934@live.co.uk

SOPHI
Structurally Optimized PHylo-Informatics.

Citation:
Chesters D (2017). Construction of a species-level tree-of-life for the insects and utility in taxonomic profiling. Syst. Biol. 66:426-439.
or
Chesters. The phylogeny of insects in the data-driven era. In submission.

Phyloinformatics pipeline optimized to three key information partitions in mined insect sequence data.
Pipeline consists of Linux commands (this document), Perl_scripts and pipeline_files.
RTF (LibreOffice) and PDF versions are provided, latter is for reading only, commands will not work in a Linux shell (due to treatment of newlines).
Available at //github.com/dchesters/sophi

Herein are given the basic set of commands used to construct constrained tree NUCL-MT-SP. Pipeline is provided in document format as to allow easier user modifications. Commands are given in blue boxes, for each of these, modify if necessary, highlight the whole command, copy and paste into the Linux shell. Other text gives instructions and further information.
Pipeline is organized into three sections, section one deals with nuclear transcriptomes, section two mitogenomes, and section three species-rich markers.

The software listed below perform key steps and will require installation. There are other less essential software which are detailed later.
Blast+ (Camacho et al. 2009) tools makeblastdb, tblastn, blastp. Available from ftp.ncbi.nlm.nih.gov/blast/executables/blast+/

Mafft (Katoh et al. 2002), from http://mafft.cbrc.jp/alignment/software/
FastTree (Price et al. 2009), from http://www.microbesonline.org/fasttree/
PhyloTreePruner (Kocot et al. 2013), from http://sourceforge.net/projects/phylotreepruner/
Raxml (Stamatakis 2014), from http://sco.h-its.org/exelixis/web/software/raxml/index.html
Sumtrees, part of DendroPy Python package (Sukumaran et al. 2010) from http://www.dendropy.org/
Usearch v4.2.66 (Edgar 2010) from http://www.drive5.com/usearch/download.html
Clustal Omega (Sievers et al. 2011) from http://www.clustal.org/omega/
BioPerl, from www.bioperl.org
Transdecoder, from github.com/TransDecoder/TransDecoder/wiki
Hamstr, from sourceforge.net/projects/hamstr/

Required files, these should usually be in working directory:
Core orthologs from Misof et al. (2014)., from: datadryad.org/resource/doi:10.5061/dryad.3c0f1
The NCBI taxonomy database, download from ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz and unzip. The two required files are named 'names.dmp' and 'nodes.dmp'. Other files can be deleted.
File 'H03InsProf.muscle.fas', these are insect profile sequences for the COI barcode region. Based on appendix B of Hebert et al. (2003). ~100 sequences aligned and trimmed to 624 bp.
File '12S_profile', aligned profile sequences for 12S rDNA.
File 'InsMito_sumtrees.procd', example mitogenome backbone topology, used when constraining barcode tree.
File 'treePL_config_file', configuration file giving settings for running treePL on species-level tree.

19 required Perl scripts.
calculate_branchlength_variance.pl, concatenate_v2.pl, create_fasta_database_from_genbank_flatfiles.pl, download_tsa_summaries.pl, exclude_parsimony_uninformative.pl, filename_processing.pl, format_conversion.pl parse_hamstr_results.pl, parse_hits.pl, parse_ncbi_deflines_fasta.pl, parse_ncbi_tax_database.pl, parse_ortholog_results.pl, parse_phylobayes_results.pl, parse_taxon_from_fastafile.pl, parse_translations_from_genbank_flatfile.pl, preprocess_fasta_database.pl, process_orfs.pl, species_filter.pl, taxon_parse_genbank_summary_results.pl

Change Log
2016-Apr-14 version 1.01: First release
2016-Apr-18 version 1.02: Some changes in order and number of steps. Extra instructions.
2017-Jan-16 version 1.03: Added quick functions section.
2019-Sep-23 version 2.01: Improvements based on new study. Proper ortholog inference. Use of Misof (2014) core orthologs. Expanded transcriptome use. Constraint functions improved and transferred to separate repository.

Section 1. Nuclear Transcriptomes

Change to your working directory:
Copy this command (with newline at the end) into the shell to run it.
Reminder! do not try to run these commands from the pdf version of this document. Use the rtf version.

```
cd [working_folder]
```

STEP 1.1
Download transcriptomes. Check current organization of the data on NCBI because directory structure changes. At time of writing this will obtain all the indices.

```
rm index.html*
wget ftp://ftp.ncbi.nlm.nih.gov/genbank/tsa/G/
wget ftp://ftp.ncbi.nlm.nih.gov/genbank/tsa/H/
wget ftp://ftp.ncbi.nlm.nih.gov/genbank/tsa/I/
```

Copy all this into a plain text file and save as tsa_list.
Count how many sequence files with:

```
grep "fsa_nt.gz" tsa_list | wc -l
```

There should be several thousand. Although these are just the file names. Note that assemblies have 4 letter designations. Next we will retrieve summary information for each transcriptome. The following Perl script is run in 2 parts. The first reads the list just created and downloads NCBI summary file for each transcriptome.
During second run each downloaded summary file is unzipped, taxon parsed, and if matching user specified taxon, records it. Script is currently set to obtain a single transcriptome for each genus. It will take the representative with the highest read count. The output is a list of download commands for the genus-filtered transcriptomes of the specified taxon, which will be run from the command line.

```
perl download_tsa_summaries.pl tsa_list 1 Hexapoda
perl download_tsa_summaries.pl tsa_list 2 Hexapoda
chmod +x transcriptome_download_commands_genusfiltered
./transcriptome_download_commands_genusfiltered
```

Transcriptome assemblies will now be downloaded, this will take some time.
You probably will need to obtain some outgroup sequences also here, the following 2 commands should help you finding suitable outgroups. For example Speleonectes is a common one for insects.

```
zgrep "Speleonectes" *
wget ftp://ftp.ncbi.nlm.nih.gov/genbank/tsa/G/tsa.GAJM.*.fsa_nt.gz
```

Some simple processing of these transcriptome files. Many software do not accept the long complex fasta IDs that they have. Also there are some inconsistencies in labeling. This type of command runs a Perl script. The script should be in the working directory or Perl wont find it (unless you set a path).

```
list_ortholog_files=(tsa.*.fsa_nt.gz)
for i in ${!list_ortholog_files[*]}
 do current_file=${list_ortholog_files[$i]};echo current file $i is $current_file
 gunzip $current_file
done

list_ortholog_files=(tsa.*.fsa_nt)
for i in ${!list_ortholog_files[*]}
  do current_file=${list_ortholog_files[$i]};echo current file $i is $current_file
  perl parse_ncbi_deflines_fasta.pl -in $current_file -format 1
done
```

Note, better to check parse_ncbi_deflines_ERROR_LOG, it will give indications if some of your files are truncated.
Next infer ORFs and translate. We will run transdecoder. For this study I used about 450 transcriptomes, which produces 32 GB of output. This software stores the results for each transcriptome in a separate sub-directory.

```
list_transcriptomes=(tsa.*.fsa_nt.b)
for i in ${!list_transcriptomes[*]}; do file=${list_transcriptomes[$i]};echo "number:$i
file:$file"
  ~/software/transdecoder/TransDecoder-3.0.1/TransDecoder.LongOrfs -t $file
done
```

We will parse the longest ORFs using the following Perl script. Before running, please open the script in a text editor and write in the path to your transdecoder results.

```
rm *.orf
list_transcriptomes=(tsa.*.fsa_nt.b)
perl process_orfs.pl ${list_transcriptomes[*]}
```

Here you can delete all transdecoder results.

STEP 1.2
Search each transcriptome using as queries the set of core orthologs, we will use the 1KITE / Misof et al. set (see above for where to obtain these). I suggest you have commands for deleting directories, as shown below. Change according to your directory names. Also recommend changing the default e-values, which are too permissive.

```
path=[path to TSAs here]
list_transcriptomes=(tsa.*.fsa_nt.b.orf)

for i in ${!list_transcriptomes[*]}; do transcriptome=${list_transcriptomes[$i]};echo
"number:$i file:$transcriptome"
  rm -rf [path here]/tmp/
  rm -rf [path here]/hmm_search_tsa_1kite/
  rm -rf [path here]/fa_dir_tsa_1kite_DMELA_5.40/
  ~/software/hamstr/hamstr.v13.2.6/bin/hamstr.pl -sequence_file=$path$transcriptome -
taxon=$transcriptome -silent -append -representative -hmmset=1kite -refspec=DMELA_5.40 -
central -eval_hmmer=1e-20 -eval_blast=1e-20
done
```

And parse results, inferred orthologs from TSAs will be placed each into single file.

```
rm insectaNUCL*
perl parse_hamstr_results.pl -in hamstrsearch_tsa_1kite.out -out_prefix insectaNUCL -
filter_duplicates -output_id_format 3
ls insectaNUCL* | wc -l # check how many orthologs have now
```

STEP 1.3
Process putative orthologs

STEP 1.3.1
Align each putative ortholog. The first command scans directory for all ortholog files (using the shared prefix insectaNUCL), then there is a loop through all these in which Mafft is invoked. Change path/installed_name of Mafft if necessary. This step consists of many jobs each using little memory, so better to use all processors.

```
list_ortholog_files=(insectaNUCL*)
no_cpus=$(cat /proc/cpuinfo | grep processor | wc -l);count=0
for i in ${!list_ortholog_files[*]}
  do current_file=${list_ortholog_files[$i]};echo current file $i is $current_file
  /usr/local/bin/mafft --localpair --maxiterate 100 $current_file > $current_file.clo &
  let count+=1
  [[ $((count%no_cpus)) -eq 0 ]] && wait
done
```

STEP 1.3.2
Build a single gene tree for each putative ortholog. An alternative way of doing this would be a Raxml bootstrap, but that is very slow. The commands scan for aligned orthologs (prefix insectaNUCL and suffix .clo) in the working directory, then invokes Fasttree for each. Again using all available processors. Change name of Fasttree command depending on your install.

```
list_ortholog_alignments=(insectaNUCL*.clo)
no_cpus=$(cat /proc/cpuinfo | grep processor | wc -l);count=0
for i in ${!list_ortholog_alignments[*]}
  do current_file=${list_ortholog_alignments[$i]};echo $i $current_file
  fasttree_2.1.7 -slow -gamma $current_file > $current_file.orth_ft &
  let count+=1
  [[ $((count%no_cpus)) -eq 0 ]] && wait
done
```

STEP 1.3.3
Paralog screening; screen each putative ortholog with Phylotreepruner. This command will probably require modification depending on your path to Phylotreepruner. Loop through names of aligned orthologs placed into object above (list_ortholog_alignments). For each ortholog, the input to Phylotreepruner is an equivelent protein alignment and tree. The five command arguments are [input.nwk], [minimum number of taxa], [input.fas], [bootstrap cutoff], and the last option refers to how the representative sequence should be selected, with nodes outside of this threshold collapsed prior to identification of subtree. The software deletes members not conforming to the maximally inclusive subtree (putative paralogs). No interesting output files are given, it just prunes the tree and fasta file, given in a new file.

```
for i in ${!list_ortholog_alignments[*]}
```

```
  do current_file=${list_ortholog_alignments[$i]};echo $i $current_file
  java -cp ~/software/phylotreepruner/src_and_wrapper_scripts/ PhyloTreePruner
$current_file.orth_ft 5 $current_file 0.5 u
done
```

STEP 1.4
Commands are given for the four matrix reductions, each accounting for different sources of bias.

STEP 1.4.1
Phylobayes for compositional heterogeneity. Obtain the software at address http://megasun.bch.umontreal.ca/People/lartillot/www/index.htm. Requires a pruned tree for each ortholog. Phylotreepruner does output one to screen, but its a little awkward to capture it, and simpler just to infer new ones for each reduced fasta file (i.e. invoke Fasttree again here). Two commands are run for Phylobayes. These use Phylip format input, so need to convert formats (Fasta to Phylip) first, note the format_conversion script will read and write several different formats (open the script in text editor for more details).
Phylobayes settings include specification of the chain length and how it is sampled. pb runs the chain, ppred -comp initiates the compositional heterogeneity test.
In the first command line below, you will need to write in the path to the Phylobayes folder, depending on where you installed it.

```
phylobayes_path=~/software/phylobayes/phylobayes4.1b/data

list_ortholog_alignments=(insectaNUCL*.clo)
for i in ${!list_ortholog_alignments[*]}
  do filename=${list_ortholog_alignments[$i]};filename_pruned=$filename'_pruned.fa'
  perl format_conversion.pl $filename_pruned $filename_pruned.phy fasta phylip
  fasttree_2.1.7 -slow -gamma $filename_pruned > $filename_pruned.orth_ft_pruned
  $phylobayes_path/pb -d $filename_pruned.phy -T $filename_pruned.orth_ft_pruned -ncat 1 -
dgam 1 -s -f -x 1 60 phylobayesOUT.$filename
  $phylobayes_path/ppred -comp phylobayesOUT.$filename
done
```

Parse the phylobayes results. The following script scans your working directory for all the phylobayes output files which should be named with prefix phylobayesOUT and suffix clo_sample.ppht. It reads the overall p-value and z-scores. Then if one of these within the user threhold, then copies the matching alignment into a new file so these subset files can easily be found later.
For parse_phylobayes_results.pl, the first argument is the parameter by which you wish to filter (zs or pv, meaning z-score or p-value), the second is the threshold value of this parameter (read the paper for more details on this, although settings used in the study were z-score of -1.1). The script makes output files with suffex RM1, thus, delete these first if present from a previous run.

```
rm *.RM1
perl parse_phylobayes_results.pl zs -1.1
```

Now you should have a set of new files with sufix RM1, concatenate these orthologs into a supermatrix. The key output is the Fasta supermatrix named insectaNUCL.smatrix, along with partition information files. Concatenation script requires user to set missing data character, this is inserted where a whole gene is missing for a given taxa. Missing data characters already in an input alignment are unchanged. The character '?' is appropriate for both DNA and Amino Acid (unlike character 'N', for example, which is sometimes used for missing nucleotide data), and should be accepted by most

software.

```
alignments=(*.RM1)
perl concatenate_v2.pl -missing_data_char ? -remove_accession 2 -matrices ${alignments[*]}
mv current_supermatrix2 insectaNUCL.smatrix.RM1;mv partitionfile2
insectaNUCL.partitionfile.RM1;mv charsetfile2 insectaNUCL.charsetfile.RM1
```

You now have a supermatrix in which the likelihood of bias from compositional heterogeneity has been reduced, it is called insectaNUCL.smatrix.RM1.

STEP 1.4.2
Screen the tree from each ortholog, for branch-length heterogeneity (nb recently I notice a software has been published for doing something similar). This step requires R, a very widely used software environment, probably installed already, if not it can be installed automatically, for example on Ubuntu this is done with the command 'sudo apt-get install r-base r-base-dev'.
For each ortholog, a Perl script is invoked which reads all branch lengths and writes these to a table . Then an R script is invoked to read this table and calculate variance, the results of which written to the file all_branch_results.

```
list_ortholog_alignments=(insectaNUCL*.clo)
rm all_branch_results
for i in ${!list_ortholog_alignments[*]}
  do filename=${list_ortholog_alignments[$i]};echo $i $filename;
  filename_pruned=$filename'_pruned.fa';
  fasttree_filename=$filename_pruned.orth_ft_pruned
  rm table_for_R R_Variance_OUT
  perl calculate_branchlength_variance.pl $fasttree_filename
done
```

Delete file 'blh_filtered_files' if you have run this step previously.
User interaction and decisions are now required. Open R (just type the capital latter R and press return), and paste in these two commands:

```
t1<-read.table("all_branch_results", header=T, row.names=1)
quantile(t1[,2], c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9))
```

On screen you will have something that looks like this:

10%	20%	30%	40%	50%	60%
0.003253877	0.007455152	0.011961814	0.017990158	0.026135755	0.036912844

It is the 10th, 20th, 30th etc percentile for variance in branch-length heterogeneity over all orthologs. For more details read the manuscript and also Johnson et al. (2013). Select a threshold that generates a matrix of roughly the same size as in other matrix reductions. For example in the manuscript I select the 36th percentile which is 0.015320058, that value is inserted in the command below and run in R. To infer this anew you will have to run the below commands (including making of the supermatrix) several time to get the value that makes a matrix of the preferred size. Here is the command (still in R) as I used in the manuscript:

```
new_filenames <- attr(t1[(1:length(t1[,2]))[t1[,2]< 0.015320058 ],], "row.names")
write(new_filenames, file = "blh_filtered_files",ncolumns = 1, append = FALSE, sep = " ")
```

Which will make a new file containing a list of file names for orthologs within the threshold for

branch-length variance.
You can now exit R, type quit(), don't need to save workspace image.
The following script reads the list of selected orthologs, then changes the name of these only, so they can be read for creation of a supermatrix.

```
rm *.RM2
perl filename_processing.pl blh_filtered_files
alignments=(insectaNUCL*.clo_pruned.fa.RM2)
perl concatenate_v2.pl -missing_data_char ? -remove_accession 2 -matrices ${alignments[*]}
mv current_supermatrix2 insectaNUCL.smatrix.RM2; mv partitionfile2
insectaNUCL.partitionfile.RM2; mv charsetfile2 insectaNUCL.charsetfile.RM2
```

Supermatrix accounting for branchlength heterogeneity is now done, named insectaNUCL.smatrix.RM2.

STEP 1.4.3
Matrix reduction with Mare. Obtain software from
https://www.zfmk.de/de/forschung/forschungszentren-und-gruppen/mare
This is conducted on a concatenated supermatrix. The input files required are the supermatrix and a character set file, latter being output from the concatenation script. As for Mare settings, a higher taxon weight is used (-t 2) so all taxa are likely to be retained. The advantage of this is that the trees from each matrix reduction method are directly comparable. Also, -d is the information weight parameter, this can be adjusted to return a matrix of more preferred size. The output is put into a new folder by Mare, so this is moved to working directory by the last command in the set below.

```
alignments=(insectaNUCL*.clo_pruned.fa)
perl concatenate_v2.pl -missing_data_char ? -remove_accession 2 -matrices ${alignments[*]}
mv current_supermatrix2 insectaNUCL.smatrix8; mv partitionfile2 insectaNUCL.partitionfile8;
mv charsetfile2 insectaNUCL.charsetfile8
mare_v01.2 insectaNUCL.charsetfile8 insectaNUCL.smatrix8 -t 2 -d 1.9 > MARE_LogScreen
mv ./results/insectaNUCL.smatrix8_reduced .
```

Supermatrix processed with Mare is called insectaNUCL.smatrix8_reduced.

STEP 1.4.4
Gblocks (Castresana 2000), from http://molevol.cmima.csic.es/castresana/Gblocks.html (popular alternative at this step would be Aliscore). The key options are: -b1= Min For A Conserved Position; -b2 = Minimum For A Flank Position; -b3 = Max Contiguous Nonconserved; -b4 = Min Length Of A Block. The unprocessed supermatrix has already been made above, so just run Gblocks here. The screen output tells you how much of the alignment was filtered.

```
Gblocks_0.91b insectaNUCL.smatrix8 -t=p -b1=19 -b2=21 -b3=8 -b4=14 -b5=a
```

STEP 1.5
Run a Raxml bootstrap. As an example, commands are provided for doing this on the first reduced matrix (insectaNUCL.smatrix.RM1).
Prior to tree search you can further process these matrices if you wish, the following script removes ambiguous, invariant, and parsimony uninformative sites.

```
perl exclude_parsimony_uninformative.pl -in insectaNUCL.smatrix.RM1 -out
insectaNUCL.smatrix.RM1.epu -data_type prot -remove_invar -remove_ambig
```

The protein model is set to BLOSUM62, although you can find the best fit model with option -m PROTGAMMAAUTO. Command might need modifying if you have Raxml installed under a different name, and of course change the names to run this on the other reduced matrices (.RM[2/3/4]).

```
perl format_conversion.pl insectaNUCL.smatrix.RM1.epu insectaNUCL.smatrix.RM1.phy fasta
phylip
mpirun -n 4 raxmlHPC-7.2.8-ALPHA-MPI -s insectaNUCL.smatrix.RM1.phy -n insectaNUCL.boot.RM1
-m PROTCATBLOSUM62 -c 4 -f a -x 12345 -p 12345 -# 250
```

The single thorough search can be demanding, you can just summarize the bootstrap trees instead (see Felsenstein 1985). Bootstrap summary also allows combination of boot trees from the different reduced matries, into a single consensus topology. Here is the command just for summary of RM1 boot trees (to combine all, just place boot trees from each of the four reduced matrices into a single file, and input into this program).

```
sumtrees.py --min-clade-freq=0.0 --log-frequency=10 --to-newick --replace --support-as-
labels --burnin=0 --output=sumtreesRM1 RAxML_bootstrap.insectaNUCL.boot.RM1
```

You might want to run the software Partition finder, to optimize the model partitions (create ‘meta-partitions’). Check the Partition finder documentation for this, though you will need to create a configuration file which contains Raxml-style partitions, as output by the concatenation script above, also a few other details. The config file will look something like this:

```
  alignment = insctNUCL.smatrix.RMmulti.epu.phy;
  branchlengths = linked;
  models = all_protein;
  model_selection = aicc;
  [data_blocks]
  gene0_insectaNUCL_EOG50000K_clo_pruned_fa_RMmulti = 1-587;
  …. several hundred partitions here ….
  [schemes]
  search = rcluster;
```

The command for running partition finder follows. It is a memory intensive step. I suggest to reduce default rcluster-max.

```
python ~/software/partition_finder/partitionfinder-2.1.1/PartitionFinderProtein.py --raxml
--processors 30 --rcluster-max 2000 partition_finder
```

Section 2; Mitochondrial Genomes

STEP 2.1

First job is to get profiles for each mitochondrial gene, these profiles will be used later for extracting homologs. These stages are simplified a little, to avoid repetition of similar jobs. Specifically, while refseq taxa are a good choice for the profiles, they require duplication of some manual steps so this is

not done. And it might be nice to add partial mitogenomes where complete mitogenomes are not available. This is a little involved also, so this step is not shown here.
Do a text search of NCBI. go to http://www.ncbi.nlm.nih.gov/, search 'nucleotide' using these terms:

```
("complete" [TITLE] AND "genome" [TITLE] AND ("mitochondrial" [TITLE] OR "mitochondrion"
[TITLE])) AND insecta [organism]
```

This will return in the region of a couple of thousand entries. Dont select one! All need obtaining, click send to file, format summary, create file. Copy results file to your working directory, the file is probably called 'nuccore_result.txt'.

STEP 2.2
Run this script, it reads the summary from genbank, and selects one entry per order, and prints the accessions for these. For this step to work, you will need in the working directory the two NCBI taxonomy database file names.dmp and nodes.dmp.

```
perl taxon_parse_genbank_summary_results.pl nuccore_result.txt 6960 1
```

The screen output has a list of accessions (one per order, so about 30 for insects), copy this list into the NCBI nucleotide search box (http://www.ncbi.nlm.nih.gov/). Click send to file, format genbank full, create file. Save the file as insecta_refseq_mtgenome_order.

Make profiles from these mtgenomes. A profile is a set of sequence which are representative of one marker. The script parses these by reading the translations.

```
perl parse_translations_from_genbank_flatfile.pl -in insecta_refseq_mtgenome_order -out
insecta_refseq_mtgenome_order.prot -out_suffx InsMT -id_format 5 -outfile_by_annotation -
preferred_annotation_field 1
```

Now in the folder there should be a file (InsMT.[gene label]) for each of the 13 PCGs of the mitogenome. The names are abbreviated.
Curation is desirable here: there might be extra files due to naming inconsistencies (identifiable through they probably have longer names, and will be very few sequences in the file). You must delete these (note this is much easier and less ambiguous on a small number of entries), which will leave exactly 13 files.

STEP 2.3
This script will be run again, to initiate obtaining all data to be used in the mitogenome analysis, not just subset to be used as profiles. This time to get a list of accessions for representatives of each insect genus.

```
perl taxon_parse_genbank_summary_results.pl nuccore_result.txt 6960 3
```

The screen output has a list of accessions (several hundred this time), copy this list into NCBI nucleotide search box (http://www.ncbi.nlm.nih.gov/), click send to file, format genbank full, create file. Save the file as insecta_mtgenome_genus.
You should add outgroups here (Diplura, Protura, Collembola), if you dont, later constraint analyses probably wont work. Also add an partial mtgenome entry from an order which currently doesnt have a

complete mtgenome (Grylloblattodea).
Copy these four accession into the NCBI nucleotide search box and download complete genbank entries: NC_022674.1,NC_026666.1,NC_010533.1,DQ241796
Manually append these four entries to the file insecta_mtgenome_genus.

Parse the .gb file for translated protein sequences, making a new fasta format file. Since protein IDs can be repeated, they are unuseful for sequence labels here, so arbitrary numbers are inserted for sequence IDs (-id_format 4).

```
perl parse_translations_from_genbank_flatfile.pl -in insecta_mtgenome_genus -out
insecta_mtgenome_genus.prot -out_suffx NULL -id_format 4 -preferred_annotation_field 1
```

Running that script under these settings outputs a single file containing all protein translations from the several hundren insects.

STEP 2.4
Make a Blast searchable database, then search it using the profiles created earlier. A note on Blast settings, the option -max_target_seqs when set to 100 or less, will probably miss sequences from outgroups, so it is set to 1000 in this case. Again, translated protein sequence is specified in tabular output, to be parsed in subsequent step.

```
makeblastdb-2.2.28+-64bit -in insecta_mtgenome_genus.prot -dbtype prot -parse_seqids
rm *InsMT_hits InsMT.allhits
InsMT_proteins=(InsMT*)
for i in ${!InsMT_proteins[*]}
  do current_file=${InsMT_proteins[$i]}; echo $i current mito gene $current_file
  blastp-2.2.28+-64bit -task blastp -db insecta_mtgenome_genus.prot -query $current_file -
out $current_file.InsMT_hits -max_target_seqs 1000 -evalue 1e-20 -outfmt '6 qseqid sseqid
evalue pident length sseq'
done
```

Parse the results of Blasting the mitogenomes, making a file for each gene containing sequences hit. A string (InMT.) is added to start of each ortholog ID.

```
cat *InsMT_hits > InsMT.allhits
rm InMT*
perl parse_ortholog_results.pl -in InsMT.allhits -out_prefix InMT -filter_duplicates -
output_id_format 3
```

STEP 2.5
Process each PCG

STEP 2.5.1
Multiple sequence alignment of each protein coding gene of the mitogenome. This step is slowish, so run on multiple cpu's. Files to be used as Mafft input as recognized through prefix 'InMT'.

```
no_cpus=$(cat /proc/cpuinfo | grep processor | wc -l);count=0
rm InMT*.clo*
split_query_files=(InMT*)

for i in ${!split_query_files[*]}
  do current_file=${split_query_files[$i]}; echo $i $current_file
```

```
  /usr/local/bin/mafft --localpair --maxiterate 100 $current_file > $current_file.clo &
  let count+=1
  [[ $((count%no_cpus)) -eq 0 ]] && wait
done
```

STEP 2.5.2
Make a single gene tree for each mitochondrial PCG alignment.

```
rm InMT*.clo.SGT
alignment_files=(InMT*.clo)
for i in ${!alignment_files[*]}
  do filename=${alignment_files[$i]};echo current file $i is $filename;
  fasttree_2.1.7 -slow -gamma $filename > $filename.SGT
done
```

STEP 2.5.3
Screen for parlogs. Step is included here, although might not be essential depending on settings used during step 2.6 (can't hurt to run it anyway).

```
rm *.clo_pruned.fa
for i in ${!alignment_files[*]}
  do current_file=${alignment_files[$i]}; echo current file $i $current_file;
  java -cp ~/software/phylotreepruner/src_and_wrapper_scripts/ PhyloTreePruner
$current_file.SGT 5 $current_file 0.5 u
done
```

STEP 2.6
Concatenate mitochondrial genes into a supermatrix. Then convert to Phylip format ready for analysis.

```
pruned_alignments=(InMT*.clo_pruned.fa)
perl concatenate_v2.pl -missing_data_char ? -remove_accession 2 -matrices $
{pruned_alignments[*]}
mv current_supermatrix2 InMT.smatrix; mv partitionfile2 InMT.partitionfile; mv charsetfile2
InMT.charsetfile
perl format_conversion.pl InMT.smatrix InMT.smatrix.phy fasta phylip
```

At this point, we are ready to make a mitogenome phylogeny constrained according to transcriptome topology. Constraint method is moved to a separate repository. Please see github.com/dchesters/treestake.

Section 3: Species-Level

Some of the initial steps of section 3 were first developed in Papadopoulou et al. (2014) and Ji et al. (unpublished).

STEP 3.1
First obtain all invertebrate nucleotide data (the 'INV' release) from NCBI. A simple command to do this is

```
wget ftp://ftp.ncbi.nih.gov/genbank/gbinv*.seq.gz
```

although in the situation that downloading is interrupted, its easier to restart if you use the following command (At time of writing there were 132 files consisting the INV release, you need to visit the site, see how many currently, then write that number in).

```
for i in {1..132};do echo "downloading file $i";
  wget ftp://ftp.ncbi.nih.gov/genbank/gbinv$i.seq.gz
done
```

The next script scans the working directory for files with names starting gb. For each one, the file is unzipped and parsed, and its fasta format sequences added a single database. Option -outformat 1 means new fasta IDs consist of >[accession]_[NCBI_taxon_number]. Note some very short sequences are ignored during this parsing stage.

```
perl create_fasta_database_from_genbank_flatfiles.pl -out inv -outformat 1
```

The next script parses the NCBI taxonomy database, for taxa descended from the node supplied by the user, in this case insects (which have the NCBI taxonomy number 50557). Only species names are given to taxonomic nodes at sub-specific ranks.

```
perl parse_ncbi_tax_database.pl -starting_node 50557 -ignore_subspecies -parse_species_only
```

Consolidate the taxonomy file and the fasta database (attach species names according to NCBI codes). The output will be called inv.parsed, and will contain only fully named insects.

```
perl parse_taxon_from_fastafile.pl -fasta_db inv -taxa_key key_Oct2013_Insecta -id_format 3
-memory_efficient -binomials_only
```

After the above step is completed, you might want to now remove file probably called 'inv', being a very large file no longer required.

Initial processing of the database. Specify limits on sequence length, any outside this range are ommited. Identical sequences are removed if they are of the same species, whereas seqeuces identical and from different species are retained (this is common in conserved and widly used markers such as 28s rRNA). This step uses usearch version 4.2.66.

```
perl preprocess_fasta_database.pl -in inv.parsed -filter_seq_length_outliers -
lower_length_limit 200 -upper_length_limit 32000
perl preprocess_fasta_database.pl -in inv.parsed.ng -reduce_redundency -usearch_command
usearch4.2.66_i86linux32
```

Finally, make a Blast searchable database of the insect sequences. Depending on memory availablity and usage during suseqeucnt search steps (particularly for COI), you can split this into several databases by reducing number in option -max_file_sz.

```
makeblastdb-2.2.28+-64bit -in inv.parsed.ng.rr -dbtype nucl -parse_seqids -max_file_sz
60000000
```

STEP 3.2

A local insect database is now ready. Next the database is searched using query profiles. The commands for making barcode-like profiles (COI, 16s rDNA, CytB etc) are NOT given here (see Chesters and Zhu 2014 for one method in objective inference of these). However, two curated profiles are provided in the supplement, COI barcode in a file named H03InsProf.muscle.fas (based on Hebert et al. 2003), and the file 12S_profile. Any extra markers you have can be added to the phylo_markers object. Also required is an alignment (between profile query and candidate sequence from the database) length cutoff, hits shorter than which will be ignored. These values go in the length_cutoffs object prior to running commands. Steps consists of Blast nucleotide search followed by Blast parsing. Arguments for parse_hits are as follows: 1) input_database, 2) input_blast, 3) percent_identity_cutoff, 4) sequence_trim_setting, 5) sequence_length_cutoff, 6) blast_output_parse_column, 7) blastdbcmd command.

```
phylo_markers=(H03InsProf.muscle.fas 12S_profile)
length_cutoffs=(500 250)

for i in ${!phylo_markers[*]}
do current_file=${phylo_markers[$i]};current_cutoff=${length_cutoffs[$i]};
  echo file $i is $current_file cutoff $current_cutoff
  rm $current_file.blastout $current_file.blastout.retreived
  blastn-2.2.28+-64bit -task blastn -db inv.parsed.ng.rr -query $current_file -out
$current_file.blastout -word_size 10 -perc_identity 40 -dust no -strand both -evalue 1e-6 -
num_threads 1 -max_target_seqs 200000 -outfmt '6 qseqid sseqid evalue pident length sstart
send qframe sframe'
  perl parse_hits.pl inv.parsed.ng.rr $current_file.blastout 60 1 $current_cutoff 2
blastdbcmd-2.2.28+-64bit
done
```

STEP 3.3
For each marker, filter and align.
Filter the sequences so that only one sequence per species is retained. There are different options for how to do this. Under the sophi default (-filter_method 2), for each species, retain the sequence with the least ambiguous sites. Two alternatives are take longest sequence for each species (-filter_method 0) or take the most representative sequence for each species (-filter_method 1). The latter works similar to the algorithm in BlastAlign (Belshaw and Katzourakis 2005), and requires an aditional script (most_representative_seq.pl). Still, I recommend -filter_method 2 in this instance, since ambiguous sites can be one source of alignment error under certain software.

```
for i in ${!phylo_markers[*]};do
   current_file=${phylo_markers[$i]}; echo file $i is $current_file;
   perl species_filter.pl -in $current_file.blastout.retreived -format 1 -
identified_species_only -filter_method 2
done
```

Before alignment you need to insert outgroups, although its not essential to do this for every single gene. In the very least, you will need three outgroups for COI corresponding to those used in the backbone trees. Here are accession information for three reasonable outgroups for COI barcode:
gb|GQ215461.1|:45-615 Cryptopygus antarcticus (Collembola);
gb|JN990600.1|:1452-2075 Occasjapyx japonicus (Diplura)
gb|JQ728012.1|:1933-2550 Acerentomon microrhinus (Protura).
These sequences would need appending to the file H03InsProf.muscle.fas.blastout.retreived.ID_filtered. Make sure the format is consistent for the outgroups, it should be >[genus]_[species]_[accession or GI].

Now to multiple sequence alignment, one of the most obviously error prone and inconsistent steps, you will probably need to try several methods for each marker. Below are the commands for each method. First select one of the following input files, or any other you might have.

```
current_unaligned_marker=H03InsProf.muscle.fas.blastout.retreived.ID_filtered
current_unaligned_marker=12S_profile.blastout.retreived.ID_filtered
```

Clustal Omega is probably the fastest and most robust multiple sequence aligner. Basic invokation follows. An alternative way to run this software would be to input a profile like this --profile1= H03InsProf.muscle.fas. The program would then convert the profile to a HMM which is used for aligning the subjects.

```
clustalo-1.2.0 --infile=$current_unaligned_marker --outfile=$current_unaligned_marker.clo --
outfmt=fa --threads=4 --force --log=clustalo_logfile
```

Mafft. There are several ways Mafft can be run, although the software rejects iterative refinment for ~50000 COI sequences. The quickest is '--retree 1 --maxiterate 0 --nofft --parttree'. Mafft alignments are characteristicly gappy, which is undesirable for length variable markers such as COI.

```
/usr/local/bin/mafft --retree 2 --legacygappenalty $current_unaligned_marker >
$current_unaligned_marker.mafft
```

Pynast (Caporaso et al. 2010) is quick, has low memory requirements, and the result is largly free of major errors. Unfortunately it often crashes, with incomprehensible error messages. Pynast conducts a Blast followed by pairwise alignment of subject seqeunces against a reference alignment. In the current instance the unaligned subjects can be aligned against the barcode profile. Note, profiles are not included in the output (which is convenient). Name of the file containing the COI profile is given in the command, change this profile name if you are using other markers.

```
pynast --min_pct_id=60 --min_len=500 --pairwise_alignment_method muscle -i
$current_unaligned_marker -t H03InsProf.muscle.fas
```

Mothur (Schloss et al. 2009). Commands to run this below, although the software makes some major errors (some sequences completely frameshifted). Again, this software aligns against a reference profile.

```
mothur "#align.seqs(candidate=$current_unaligned_marker, template=H03InsProf.muscle.fas,
match=2.0, mismatch=-2.0, gapopen=-4.0, search=kmer, align=needleman, threshold=0.7)"
```

BlastAlign (Belshaw and Katzourakis 2005). Always remarkable error free, although with the property that many fragments are omitted in the output. The program can use a single reference sequence for formation of the MSA, or if the user does not supply one, the software will search for the most representative sequence, in a sample of the input file. A newer version of the software runs on Blast+, whereas older versions use legacy Blast.

```
perl BlastAlign -i $current_unaligned_marker
```

STEP 3.4

Concatenate a species level supermatrix. A key setting when concatenating this data, is whether or not to retain species depending on which markers are present. The default behavious is to retain species only if they have a COI, with presence of other loci ignored. This is since the main use of this tree was in tree based DNA barcoding, which is based on COI. This setting is applied with the option -required_data Y? meaning first gene is required for printing a concatenate (Y), concatenate is printed irrespective of presence of second gene (?), as long as other specifications are met.
In the first line, write the names of the aligned single-gene matrices which are to be concatenated.

```
list_matrices_for_concatenation=(H03InsProf.muscle.fas.blastout.retreived.ID_filtered.clo
12S_profile.blastout.retreived.ID_filtered.clo)

perl concatenate_v2.pl -missing_data_char ? -remove_accession 1 -required_data Y? -matrices
${list_matrices_for_concatenation[*]}
mv current_supermatrix2 insect8geneA; mv partitionfile2 insect8geneA.partitionfile; mv
charsetfile2 insect8gene.charsetfile;
```

STEP 3.5
Here, phylogenetic inference of species-rich data, constrained by backbone topology. Please see: github.com/dchesters/treestake.

References

Belshaw R, Katzourakis A. 2005. BlastAlign: a program that uses blast to align problematic nucleotide sequences. Bioinformatics, 21:122-123.
Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K. Madden TL. 2009. BLAST+: architecture and applications. BMC Bioinformatics, 10:1-421.
Caporaso JG, Bittinger K, Bushman FD, DeSantis TZ, Andersen GL, Knight R. 2010. PyNAST: a flexible tool for aligning sequences to a template alignment. Bioinformatics, 26:266-267.
Castresana J. 2000. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol. Biol. Evol., 17:540-552.
Chesters D, Zhu C-D. 2014. A protocol for species delineation of public DNA databases, applied to the insecta. Syst. Biol., 63:712-725.
Edgar RC. 2010. Search and clustering orders of magnitude faster than BLAST. Bioinformatics, 26:2460-2461.
Felsenstein J. 1985. Confidence limits on phylogenies: An approach using the bootstrap. Evolution 39: 783-791.
Hebert PDN, Ratnasingham S, DeWaard JR. 2003. Barcoding animal life: Cytochrome c oxidase subunit 1 divergences among closely related species. Proceedings of the Royal Society B: Biological Sciences, 270:S596–S599.
Johnson BR, Borowiec ML, Chiu JC, Lee EK, Atallah J, Ward PS. 2013. Phylogenomics resolves evolutionary relationships among ants, bees, and wasps. Current Biology, 23:2058–2062.
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.
Kocot KM, Citarella MR, Moroz LL Halanych KM. 2013. PhyloTreePruner: A phylogenetic tree-based approach for selection of orthologous sequences for phylogenomics. Evolutionary Bioinformatics, 9:429-435.

Papadopoulou A, Chesters D, Coronado I, De la Cadena G, Cardoso A, Reyes JC, Maes JM, Rueda RM, Gómez-Zurita J. 2014. Automated DNA-based plant identification for large-scale biodiversity assessment. Molecular Ecology Resources, 15:136-152.
Price MN, Dehal PS, Arkin AP. 2009 . FastTree: computing large minimum-evolution trees with profiles instead of a distance matrix. Molecular Biology and Evolution, 26:1641-1650.
Schloss PD, Westcott SL, Ryabin T, Hall JR, Hartmann M, Hollister EB, Lesniewski RA, Oakley BB, Parks DH, Robinson CJ, Sahl JW, Stres B, Thallinger GG, Van Horn DJ, Weber CF. 2009. Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol., 75:7537-41.
Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Soding J, Thompson JD, Higgins DG. 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular Systems Biology, 7:539.
Stamatakis A. 2014. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30:1312-1313.
Sukumaran J, Holder MT. 2010. DendroPy: A Python library for phylogenetic computing. Bioinformatics, 26:1569-1571.