EAPhy v1.1 Manual

Exon Alignment for Phylogenetics

For Questions: mozes.blom@gmail.com

Summary

EAPhy, Exon Alignment for Phylogenetics, consists of a collection of scripts that jointly process exonic sequences, construct alignments using an existing aligner, check and filter alignments and ultimately generates the required input for commonly used phylogenetic inference programs. The main aim of this pipeline is to provide a flexible but rigorous tool to generate reliable alignments and by minimising the time to do so, promote the application of multiple species tree methods and examine the (in)congruences between distinct inference approaches (see Blom et al., in prep).

All scripts are written in Python and throughout the pipeline, many functions benefit from modules that have been developed as part of the BioPython distribution (http://biopython.org/wiki/Main_Page).

Quickstart

1. Specify the correct file/folder Path's in EAPhy_config.py script
2. Specify the analysis criteria depending on the dataset
3. Run EAPhy: $Python EAPhy_config.py
4. If required, visually inspect flagged alignments in folder: *~/homePath/results/run_folder/ambiguous_or_haplotype/2.alignments/2.checked_alignments/2.alignments_to_check/alignments_require_inspection.txt* Save modified alignments in the */4.alignments_filtered/* folder and append exact alignment names to */4.alignments_filtered/alignments_filtered.txt* file.
5. Re-run the parts of the pipeline, subsequent to the alignment_check step (Note: make sure that if you rerun the pipeline the original output folders for those parts of the pipeline are renamed or removed. I.e. If you only rerun the concatenation step, then make sure to manually rename the concatenation folder or move it out of the results directory).

NOTE: EAPHY ASSUMES MOST SEQUENCES ARE CONTINUOUS, IF SOME INDIVIDUALS CONTAIN LARGE AMOUNTS OF MISSING DATA ('N's) WITHIN SEQUENCES, PARAMETER SETTINGS NEED TO BE ADJUSTED APPROPRIATELY!!

Overview

EAPhy was developed to enable the rapid conversion of exon-capture data, into reliable alignments for population or phylogenetic inference. The premise of phylogenetic inference depends on comparison of orthologous sites and EAPhy was designed to utilise existing aligners and inspect alignment quality in a high-throughput way. It is specifically constructed to use the coding codon character of exons, by converting the nucleotide sequence alignments into amino acid alignments that are subsequently cleaned up and inspected for anomalous patterns, following a set of criteria that can be adjusted based on the dataset. EAPhy is not designed to identify individual sequencing errors that are often associated with Next-Generation Sequencing (NGS) datasets, but will pinpoint sequence regions with an excess amount of non-synonymous substitutions (potential 'low-coverage' sequences) if these have not been filtered out prior and appear anomalous in the resulting alignments.

In addition to generating high quality alignments, the second objective of EAPhy is to reduce the bioinformatic complexity associated with analysing NGS data by providing one tool to generate the input data for several distinct phylogenetic methods. EAPhy will create both sequence based alignments, haplotype or diplotype, and SNP based alignments. Thus generating the input for programs such as PhyML, RAxML, Beast, *Beast and SNAPP. Inferred gene trees can subsequently be used in summary statistic methods for species tree inference, such as MP-est or Astral. Important to note is that since alignment filtering is conducted by amino-acid position, EAPhy will identify the correct codon position for each nucleotide position and can thus account for codon position during phylogenetic inference by creating data partitions using PartitionFinder.

Finally, EAPhy can generate alignment datasets with various levels of missing data. It is currently not well understood how missing data might affect phylogenetic inference and by using datasets with different levels of missing data, this can be further explored for each specific dataset.

Running EAPhy

Installation

EAPhy has been developed using Python 2.7, so my advice is to use this distribution, which should already be pre-installed on most Mac's (from OSX Lion, 10.7, onwards) but should be installed on other platforms if not done prior. The software was developed for usage on a Macintosh (Mac OSX Mavericks, 10.9) and other Unix based systems. It has not been tested for Windows based systems and my suggestion is to either use a Mac or Linux machine, or a virtual machine (e.g. Parallels).

EAPhy is dependent on the following Python modules:

Bio
os
matplotlib
csv
subprocess
random
pylab

If you are a user from an educational institution, my suggestion is to install the Enthought Canopy distribution of Python (https://www.enthought.com) which comes preloaded with a large number of useful modules for scientific computing. Installing additional modules, such as listed above, is pretty straightforward and will seem familiar to users of R-studio.

Lastly, an aligner needs to be compiled and the path to the binary needs to be specified, in the EAPhy config script. I have been using Muscle v.3.8.31 (http://www.drive5.com/muscle/), in principle it should be possible to use any aligner of choice, but if you are intending to use an alternative aligner, the script 'runAlignment.py' needs to be adjusted to match the appropriate Terminal call. If you are using a Mac operating system, it is easy to find the correct path to the binary by using the Terminal application or alternatively, by right-clicking on the Muscle binary —> Get info —> General —> Where. Copy this path to the EAPhy config script, followed by the name of the binary. E.g.: /Users/your_name/bin/Muscle3.8.31

To double check whether all modules are properly installed, open a Python session via the Terminal and check whether each module is installed by using:

'> import *insert_module_name*'.

If above modules and aligner are installed, everything should be in place to run EAPhy.

EAPhy Set-up

First copy the complete EAPhy folder to a destination of choice and record the Path to the folder in a similar way as described in the 'Installation' section. This will be your 'homePath', later to be specified in the EAPhy_config script. Then create the following input files in the EAPhy folder:

1. Loci folder: A folder containing all your sequenced contigs per locus, in **Fasta** format. For example, in /EAPhy/examples/test_ambig_contigs/, there are 10 loci, each locus file containing one sequenced contig for each individual. Note, the pipeline assumes that all sequences start in first frame/first codon position(!).
2. Loci list: A txt file that lists all the loci you wish to include in the analysis. Each listed locus should **exactly** match the file name in loci folder.
3. Individuals list: A txt file that lists all individuals you wish to include in the analysis. Each listed individuals should **exactly** match the fasta sequence ID within each file in the loci folder.
4. Optional: If you would like to change the fasta sequence IDs used and/or would like to have alignment files in the 'phylip-sequential' output format but have sequence IDs longer than 10 characters, you can create a tab-delimited text file with two columns (old_ID and new_ID respectively). This will change the sequence IDs in all subsequent output alignments. For example, see: '/EAPhy/examples/test_id_change_diplo.txt'.

Note: If you wish to use haplotype data instead of diplotype data, then each individual should have both haplotypes in the single locus file (see loci folder), but with each haplotype separately identified in the Individuals list. For example, see: '/EAPhy/examples/test_indivs_haplo_list.txt'

EAPhy Configuration Script

EAPhy is coordinated in such a way, that the user only has to provide run-specific parameter settings in the EAPhy_config.py script. This script calls dependable modules and scripts from within and runs the complete analysis. Python scripts can be opened with any text editor, i.e. Sublime (http://www.sublimetext.com), and can be identified by their .py extension (do not change extension!).

Input:

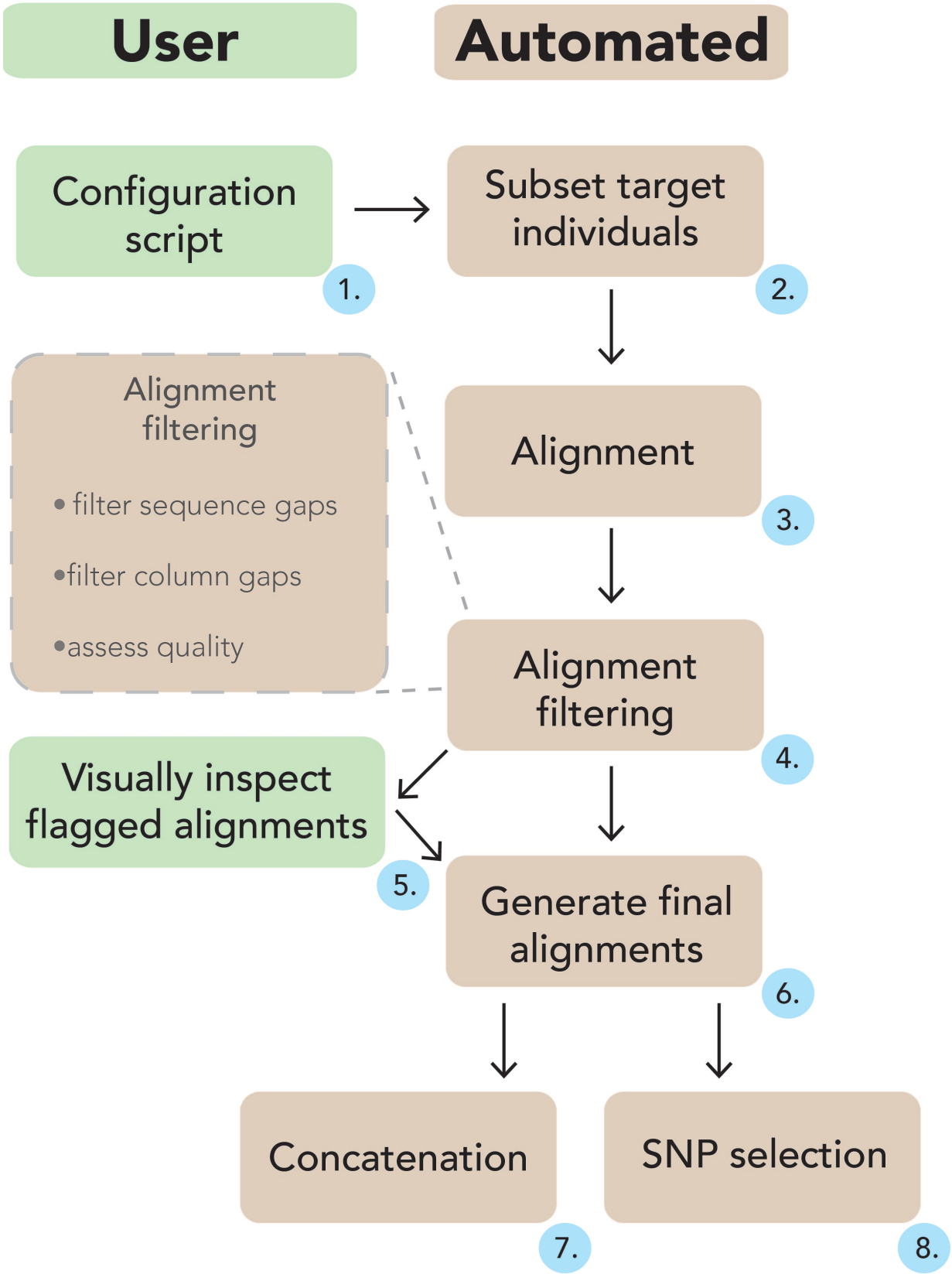
homePath	Absolute path to EAPhy folder, keep single quote marks in place
bestcontigPath	Path to loci folder. If in EAPhy folder, just specify folder name, keep single quote marks in place
individuals	Path to individuals list. If in EAPhy folder, just specify folder name, keep single quote marks in place
exons	Path to exon list. If in EAPhy folder, just specify folder name, keep single quote marks in place
out_name	Specify an output name, will be the name of your run-specific folder
path_to_scripts	Path to scripts folder, if not moved from EAPhy folder, no change needed
aligner	Path to aligner

Which analysis to run

The EAPhy analysis is compartmentalised in such a way, that the complete analysis can be run, from contigs to input alignments for phylogenetic methods, or alternatively specific parts of the analysis can be rerun. See the next page for a graphical overview of the workflow.

Specify in this section whether you want to run the complete analysis or subsets of the analysis by annotating '1' (yes) or '0' (no). Note, that each subsequent part of the analysis pipeline is dependent on the previous section. If the directory structure is not in place, consecutive components will not execute.

complete	Run the complete pipeline from start to finish
subset_indivs	Create fasta files, containing only the contigs for individuals specified in 'individuals'. This enables subsetting of datasets for specific subgroups
alignments	Generate alignments with the specified aligner
align_check	Conduct the amino-acid coding alignment checks
align_stats	Calculate some basic alignment statistics
align_final	Generate the final alignments, using the individuals that are specified in 'indivs'. This enables subsetting of individuals without having to rerun the alignments. If the same individuals need to be included then 'individuals' equals 'indivs' (for 'indivs' see 'Number of missing indivs allowed')
concatenation	Generate concatenated alignments including all alignments listed in the '3.final_alignments' folder, for each value of missing individuals allowed. A Partition Finder input file is also generated, specifying loci and codon position within the concatenated alignment. Note that a minimum alignment cut-off length can be specified (see 'Minimum alignment length'). Haplotype based sequence data are not concatenated.
snp_selection	Generate alignments where either ALL or ONE snp per gene are concatenated. SNPs can be enforced to be only considered when biallelic, i.e. as required for a SNAPP analysis. Random SNP selection can be repeated to test for inconsistency of phylogenetic estimation due to random SNP selection.


User
Automated
Configuration script
1.
Subset target individuals
2.
Alignment
3.
Alignment filtering
4.
Alignment filtering
• filter sequence gaps
•filter column gaps
•assess quality
Visually inspect flagged alignments
5.
Generate final alignments
6.
Concatenation
7.
SNP selection
8.

Datatype

EAPhy can align both phased haplotype data and diplotype data, with ambiguous sites specified according to the IUPAC coding system. If haplotype data is used, then both haplotypes for each individual should be included in the single fasta file for each locus. Each haplotype should have a unique Fasta sequence ID, such as 'individual_ID_h0' and 'individual_ID_h1'. Both should exactly match the individual ID's, listed in the 'individuals list'. If the datatype equals haplotype data, then concatenation and SNP selection are not performed when running the complete analysis. Lastly, the additional paralogy identification step examining locus specific average heterozygosity levels, during alignment filtering, is not included when using haplotype data.

Parameter settings

EAPhy was designed to provide flexibility in the stringency of alignment filtering, depending on different datasets, enabling its application for both population and species level exon-capture data. The parameter settings that need to be specified reflect that flexibility, the default settings were used for the analysis of a genus level species radiation and could for example be relaxed for the analysis of populations. A graphical representation of the alignment filtering is provided in the supplementary material, at the end of the manual. Note, EAPhy does assume that each contig starts in first codon-position.

max_stop	Alignments containing more than 1 stop-codon for any individual sequence will be flagged for further inspection. Note that the BioPython module also identifies codons which it cannot confidently translate as stop codons, so if a large number of N's are present within sequences the max_stop requirement needs to be relaxed otherwise many alignments will be flagged.
indiv_gap_window	Window length, in amino-acid number, used to filter out sequence gaps within individual sequences.
indiv_gap_ratio	The gap ratio (between 0 and 1) used to determine which sequence windows need to be removed. E.g. If the indiv_gap_window equals 7 and the indiv_gap_ratio 0.5, then the individual sequence window assessed is removed if 4 or more amino acid positions are not present (consist of N's).
column_gap_window	Window length, in amino-acid number, used to filter out codon columns with missing data.

column_gap_ratio The column ratio (between 0 and 1) used to determine which column windows need to be removed. E.g. If the column_gap_ratio is 0.01 and the column_gap_window is 3, then an alignment column window of three amino acids is removed (thus across all individuals) if more than 2 amino acid alignment columns have missing data. If column_gap_ratio would be 0.5, then columns that have over half of the individuals sequenced for that amino acid position are not counted as having missing data. If stringent, this removes 'jagged alignment edges' in particular. See figures for example.

max_insertion_cut_off The max-number of individuals sequenced for each nucleotide column for it to be considered an erroneous insertion and thus removed. For example, if max_insertion_cut_off equals 1, then every nucleotide column that is only sequenced once is removed from the alignment. This check is conducted after the column_gap_window and thus should not disrupt the correct codon position assignment for each nucleotide column afterwards, as long as max_insertion_cut_off is smaller than column_gap_ratio * number of individuals, it rather only removes erroneous columns.

sliding_window_div_check The sliding window length, in amino acid number, used to compare individual sequences to the inferred consensus sequence.

max_heterozygosity_ratio The max ratio of amino-acids in the sliding window that can be divergent from the consensus sequence, for it not to be flagged as an alignment that needs to be manually checked (2.alignments_to_check).

average_heterozygosity_cut_off The cut_off ratio used to flag potential paralogs when assessing diplotype data (0 = all alignments flagged, 1 = non flagged). For example, if average_heterozygosity_cut_off equals 0.98, the 2% of the loci with the highest average individual sequence heterozygosity per alignment, are flagged as potential paralogs (3.alignments_paralogy).

indivs_included Here you can specify the path to a text file, similar as the individuals file (same requirements) where you can list

	the individuals you wish to include in the final alignments. This extra list is included, so individuals can be subsetted without having to rerun all alignments and alignment filtering. If the final list with individuals is the same as the original one, then keep 'indivs_included = individuals'. A usage example: Change the indivs_included list and only repeat the analysis from the final_alignment step onwards
number_missing_indivs_allowed	A list of values, keep brackets and separate values with a comma, between 0 and total number of individuals that indicates how many final alignment subsets need to be created accounting for an x-amount of missing data. For example, 0 means that all final alignments have 0 missing individuals (see indivs_included list) and 1 means that the final alignments have 0 missing individuals or utmost 1 individual from the indivs_included_list missing. Note, only alignments that passed the alignment filtering step and are specified in the resulting '4.alignments_filtered' are considered.
id_change	Specify whether the 'sequence-id's' used in the original Fasta files require to be updated (1 = yes, 0 = no). This is required if 'phylip-sequential' is a requested output format (concatenation!!) and sequence-id's are currently more than 10 characters long. If 'id_change = 1' then a tab-delimited text file is required with each current individual name and the corresponding new individual name annotated. Or this option could also be used if you would only like to update the individual names.
id_change_list	Here you can specify the path to a tab-delimited text file, with the 'old fasta sequence id's' in one column and the newly designated sequence id's in a second column. Each column should be headed with 'old_ID' or 'new_ID' respectively, and file should be 'tab-delim'! See for example, the file 'test_id_change_diplo.txt' in the examples folder.
min_align_length	Specify the minimum alignment length, in nucleotide number, for an alignment to be included in the

	concatenation alignments. Individuals could have missing sequences, but total alignment length should at least exceed this value.
branch	Parameter setting used for PartitionFinder input file, see PF manual for detailed description
model_evol	Parameter setting used for PartitionFinder input file, see PF manual for detailed description
model_sel	Parameter setting used for PartitionFinder input file, see PF manual for detailed description
search	Parameter setting used for PartitionFinder input file, see PF manual for detailed description
biallelic	Specify (1 = yes, 0 = no) whether polymorphic sites for SNP selection are only considered as polymorphic if they are biallic. This means that SNPs can have either 1 homozygous genotype and 1 heterozygous genotype (should include homozygous allele) or 2 homozygous genotypes and 1 heterozygous genotype (should consist of both homozygous alleles). The genotype of heterozygous individuals should be coded according to IUPAC specification. This enforcing of biallelicness is in particular included since it is an assumption of the SNAPP model.
reps	Number of times the random SNP selection (one per gene) should be repeated
unique	Specify (1 = yes, 0 = no) whether only one SNP per gene should be sampled, if multiple exons of the same gene are included. SNAPP assumes unlinked sites, which is unlikely if SNPs originate from the same gene but different exons. If this requirement is enforced, the naming of the loci should have a SPECIFIC FORMAT, namely: gene_exonNumber_species.fasta (i.e. COI_exon2_Hsapiens.fasta). The gene name SHOULD be specified followed by two underscores. It will then make sure, that only SNPs are sampled from distinct genes.

Run Analysis

Once all is set, an EAPhy run can easily be initiated by opening the Terminal application (Mac OSX / Linux). To start EAPhy simply type ‘Python’ followed by a space and then drag the EAPhy_config.py script onto the Terminal or Command Prompt, hit enter and the run should initiate. The first time you run EAPhy, a results folder should appear in the directory that you specified as ‘homePath’, most likely the EAPhy folder, where all output folders are stored. Note that a new run specific folder is created, but if the folder exists and you are attempting to overwrite folders then a warning error will appear and you should either remove the existing folder or rename your new run. If you are running only parts of the pipeline, i.e. concatenation and/or SNP selection, and those folders do not exist yet, then they are created in the already existing folder. Similarly, if you run a phased haplotype version of the same dataset, then a new folder will be created called ‘haplotype’ and the ‘ambiguous’ folder will still be retained.

Output

Once you initiated the EAPhy run, a run specific (named accordingly specified in the config script) folder is created in the Results folder. Within the run specific folder, a folder is created that's either called 'ambiguous' or 'haplotype' depending on the dataset analyzed. Within that folder, all output files are stored in a consistent structure E.g.:

Results

A_run_name_specified

Ambiguous

1.best_contigs

2.alignments

3.concatenation

4.SNP_selection

1.best_contigs

This folder contains all loci, as specified by the loci list, but only including the target individuals as specified in the individuals list. These are the Fasta input files, suited for subsequent aligning.

2.alignments

This folder contains three sub-folders:

2.alignments

1.basic_alignments

2.checked_alignments

1.alignments_passed

2.alignments_to_check

3.alignments_paralogy

4.alignments_filtered

5.filtered_align_stats

3.final_alignments

1.basic_alignments:	Contains all aligned files, the Muscle output files for example.
2.checked_alignments:	Contains five sub-folders
1.alignments_passed:	This folder contains all checked and modified alignments that fulfil all filtering criteria set in the config script.
2.alignments_to_check:	This folder contains all loci that did not pass the alignment filtering and require to be visually inspected.
3.alignments_paralogy:	This folder contains the loci with the highest average heterozygosity per individual, as specified by the cut-off in the config script
4.alignments_filtered:	This folder is identical to the folder “1.alignments_passed” but is meant to be supplemented with visually inspected loci that after modification are suitable to be included in the downstream analysis. Such modified loci should be stored in the folder “4.alignments_filtered/alignments/“ and appended, exact alignment file name(!!!), to the loci list: “4.alignments_filtered/alignments_passed.txt”.
5.filtered_align_stats:	This folder contains a csv file with information regarding the loci length and number of individuals recovered for the filtered_alignments
3.final_alignments:	Contains a folder for each value of total number of missing individuals allowed. The loci_not_included text file will contain all loci that had more individuals missing for each alignment than allowed for that specific value. This folder also contains a graph which shows the relationship between number of missing

individuals allowed and the number of recovered loci (NumberOfLociRecoveredMissingData.png).

3.concatenation

This folder contains a folder for each value of total number of missing individuals allowed. All loci that are specified in the */2.alignments/3.final_alignments/ *_missing_indivs_allowed/loci_included_*.txt,* will be included in the concatenated alignment as long as they fulfil the minimum length requirement specified in the config script. In addition to a concatenated alignment in both Fasta and Phylip format, a PartitionFinder input file is created where the position of each locus and codon position in the concatenated alignment is specified.

4.SNPselection

This folder contains a folder for each value of total number of missing individuals allowed. A random SNP will be selected, given the specified criteria, from each locus that is listed in */2.alignments/3.final_alignments/*_missing_indivs_allowed/loci_included_*.txt,* if 'unique' in the config script is not specified, alternatively, only one SNP per gene is selected if multiple exons belong to the same gene. In addition, a SNP alignment is stored that contains all SNPs for all loci, given the specified criteria.

If the user specified that multiple random SNP selection rounds were to be executed, then randomly repeated sampling runs are stored as separate SNP alignments.

Post EAPhy

Once EAPhy has finalized, the user has the required input files for a range of subsequent species tree inference analyses. For example, in */2.alignments/ 3.final_alignments/*_missing_indivs_allowed/,* all required filtered alignments are present for inferring individual gene trees. The concatenation folder contains the necessary input files for running a PartitionFinder analysis to estimate optimal partitioning schemes and most appropriate substitution models, to eventually enable inference of a concatenated Maximum Likelihood tree. The folder */4.SNPselection/* contains random subsets of SNPs across loci, to run a SNP based species tree estimation method such as SNAPP. Using the summary statistic file in /2.alignments/*2.checked_alignments/5.filtered_align_stats/*, the longest loci can be selected (which are likely most informative, see REF moos) for a species tree co-estimation method such as *Beast or BEST, that currently are not able to utilize complete phylogenomic datasets due to computational complexity.

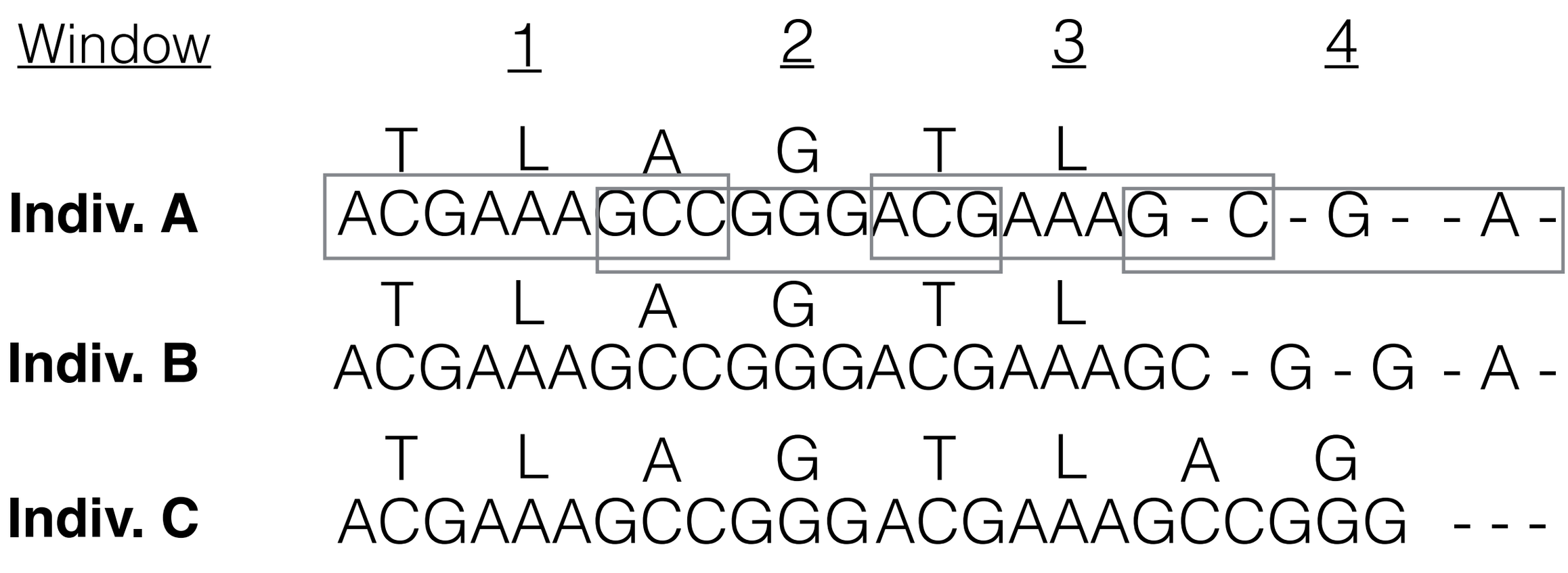
Furthermore, EAPhy also enables further investigation into the potential challenges associated with missing data. By relaxing the criteria set for alignment filtering, the relative amount of missing data within loci can be adjusted. Similarly, the amount of missing data between loci (loci completely missing certain individuals), can be accounted for and potential inferential consequences further examined.

Alignment Filtering

Alignment Filtering - 1

Criteria:	
Indiv_gap_window	3
Indiv_gap_ratio	0.5
column_gap_window	3
column_gap_ratio	0.5

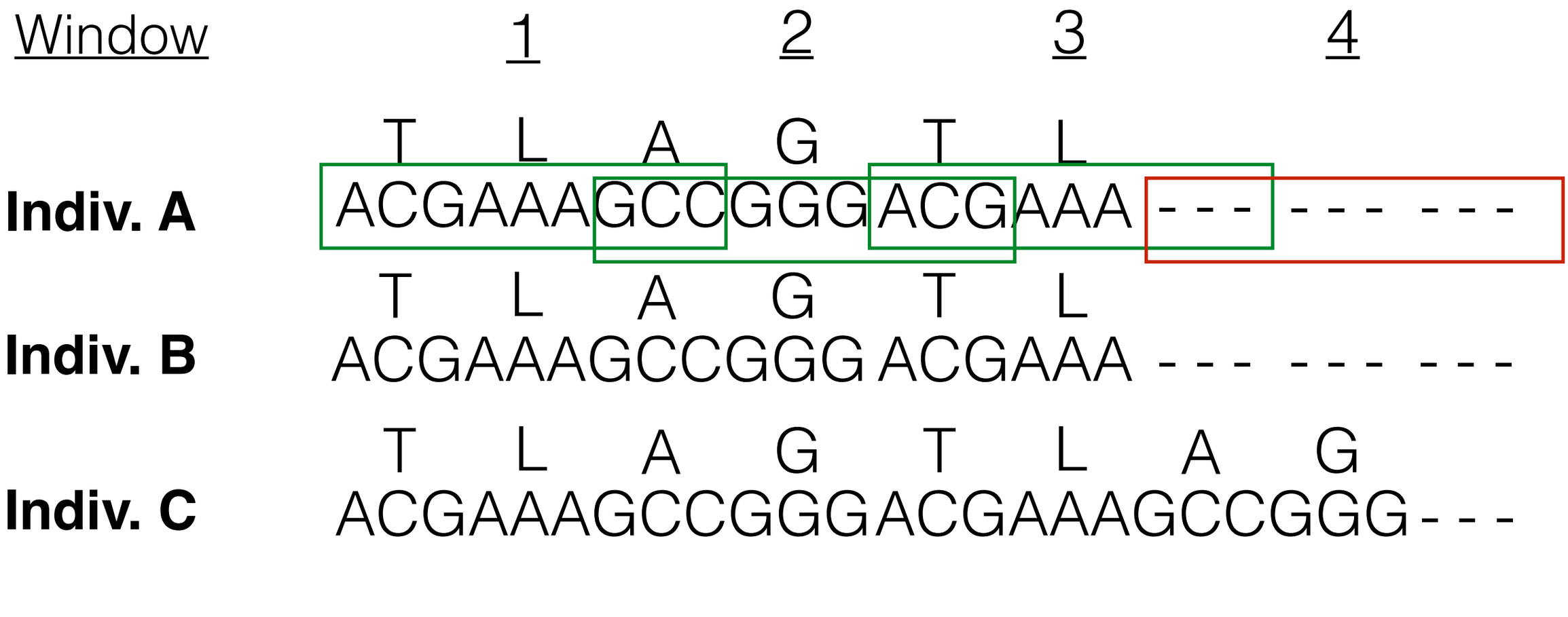
An example of alignment filtering, for a specific set of filtering criteria. Nucleotide sequence alignments are converted into amino-acid alignments. Then alignment quality is assessed in a jumping-window framework. Here window size is three and windows are indicated with brackets. After check of each window, the next window contains half of the previous codons minus one.



Alignment Filtering - 2

Criteria:	
Indiv_gap_window	3
Indiv_gap_ratio	0.5
column_gap_window	3
column_gap_ratio	0.5

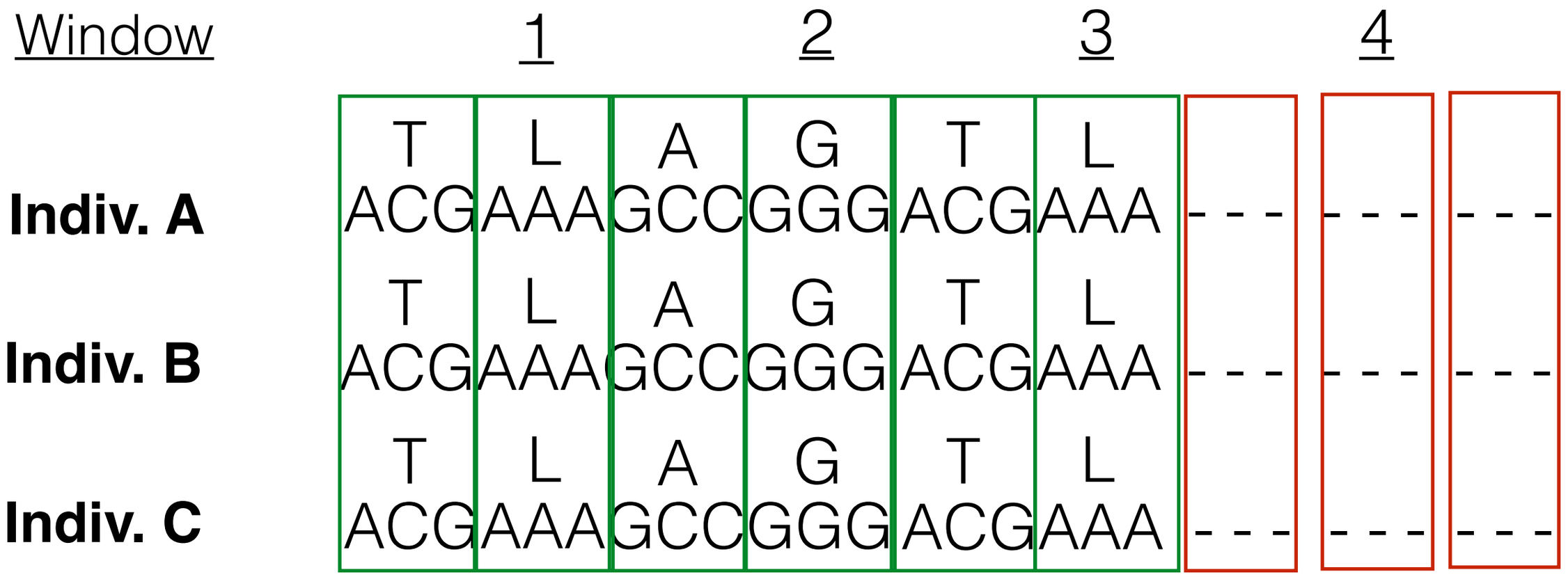
The first step is to remove gapped and/or erroneous parts of the alignment within each individual. In this case individual A and individual B have part of their sequence removed because window 4 had more than half of the sequenced codons missing. This window is retained within individual C.



Alignment Filtering - 3

Criteria:	
Indiv_gap_window	3
Indiv_gap_ratio	0.5
column_gap_window	3
column_gap_ratio	0.5

The second step is to remove alignment columns that have missing sequence data for more than a specified ratio of individuals. Here, all alignment columns for window four are removed (i.e. the first codon of Indiv. C) since more than half of the columns within window four had less than half of the individuals sequenced (column gap ratio)



Alignment Filtering - 4

Criteria:	
Sliding_window_div_check	3
max_heterozygosity_check	0.5
max_stop	1

The last step is to assess whether alignments are as expected. For each alignment, the amino acid consensus sequence is estimated and individual sequences compared to the consensus. Here they are all identical, so none of the windows have more than half of the codons distinct. Similarly, no stop codons are sequenced and thus this alignment will pass the filtering step and will not be flagged for visual inspection.