BAGpipe: pipeline for Biodiversity Assessment using Genbank data

User Manual

Anna Papadopoulou
Animal Biodiversity and Evolution
Institut de Biologia Evolutiva
(CSIC-Univ. Pompeu Fabra)
Barcelona, Spain

Douglas Chesters
Key Laboratory of Zoological Systematics and Evolution
Institute of Zoology
Chinese Academy of Sciences
Beijing, China

&

Jesús Gómez-Zurita
Animal Biodiversity and Evolution
Institut de Biologia Evolutiva
(CSIC-Univ. Pompeu Fabra)
Barcelona, Spain

BAGpipe: pipeline for Biodiversity Assessment based on Genbank data

User Manual

Contents:

Unix commands

In order to make better use of the pipeline, you will need to be familiar with some basic Unix/Linux commands. If you are not, there are several web pages that can help you with this, for example:

http://www.ee.surrey.ac.uk/Teaching/Unix/

http://www.math.utah.edu/lab/unix/unix-commands.html

http://korflab.ucdavis.edu/Unix_and_Perl/

You can always refer to the 'man' command for command usage and options. For example, in order to find out how to use the 'tar' command:

```
% man tar
```

System requirements

BAGpipe can be run on any computer under Linux or Mac OS operative systems, with a minimum of processing memory around 6-8 GB and at least 50 GB of free space for storage of data retrieved from public repositories of nucleotide sequences. For some of the pipeline steps a multi-processor system will be very advantageous.

Installing BAGpipe

The BAGpipe package is distributed as a zipped set of folders, which can be downloaded from:

http://www.ibe.upf-csic.es/SOFT/Softwareanddata.html

and

http://sourceforge.net/users/dchesters

Unzip and copy the folder anywhere in your system (for example on the Desktop or Documents folder). It is advisable to keep the original folder structure, which corresponds to the default paths given in the pipeline. All freely distributed scripts can be found in two folders: "Scripts1_database" and "Scripts2_identification". These folders contain our custom Perl scripts for different steps of the analysis. Moreover the pipeline depends on external software (listed below), which needs to be downloaded and installed independently. Note that the program versions mentioned below are just examples. In almost all cases (apart from USEARCH), you are advised to download the latest available version that is suitable for your system, and adjust the commands accordingly.

Software Installation - Linux

1. NCBI BLAST

- Go to:
  http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download
  Or connect directly as 'Guest' to:
  ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
- Copy the appropriate zipped file for your system (for example, 'ncbi-blast-2.2.28+-x64-linux.tar.gz') to the folder 'NCBI_BLAST' of the BAGpipe folder structure.
- cd to this folder and extract the downloaded zipped file. Then rename the resulting folder in order to be able to use the standard path as included in the pipeline, and clean up if you want. For example (*you need to change the program version accordingly*):

```
% cd BAGpipe/NCBI_BLAST
% tar -zxvf ncbi-blast-2.2.28+-x64-linux.tar.gz
% mv ncbi-blast-2.2.28+ ncbi-blast
% rm ncbi-blast-2.2.28+-x64-linux.tar.gz
```

2. USEARCH

- Go to: http://www.drive5.com/usearch/download.html
- Select the linux platform and the version you want to download. We currently recommend **v4.2.66**, as the pipeline has been built using this version. Newer versions will probably require some changes in the command line.
- Fill in your email address. You will immediately receive an email with a link for downloading the requested executable.
- Move the downloaded executable (usearch4.2.66_i86linux32) to the 'USEARCH' folder in the folder structure of BAGpipe.
- Change file permissions in order to be able to execute it:

```
% cd BAGpipe/USEARCH
% chmod u+x usearch4.2.66_i86linux32
```

3. EMBOSS

- Go to: http://emboss.sourceforge.net/download/
  Or connect directly as 'Guest' to:
  ftp://emboss.open-bio.org/pub/EMBOSS/
- Copy the current version of the EMBOSS zipped file (for example 'EMBOSS-6.6.0.tar.gz') to the folder 'EMBOSS' of the BAGpipe folder structure.
- cd to this folder and extract the downloaded file. Rename the resulting folder (in order to be able to use the standard path included in the pipeline). Compile the program using configure and make. For example (*change the version number accordingly*):

```
% cd BAGpipe/EMBOSS
% tar -zxvf EMBOSS-6.6.0.tar.gz
% rm EMBOSS-6.6.0.tar.gz
% mv EMBOSS-6.6.0 EMBOSS
% cd EMBOSS
% ./configure
% make
```

4. **MAFFT**

- Go to: http://mafft.cbrc.jp/alignment/software/linuxportable.html and download the portable Linux version of the program (for example, 'mafft-7.123-linux.tgz').
- Move the downloaded zipped file to the 'MAFFT' folder of the BAGpipe folder structure. Then cd to this folder, extract the zipped file and rename one of the two resulting folders ('mafft-linux64') which contain the executable mafft.bat. For example (*change the program version accordingly*):

```
% cd BAGpipe/MAFFT
% tar -xf mafft-7.123-linux.tgz
% rm mafft-7.123-linux.tgz
% mv mafft-linux64 mafft
```

5. **RAxML**

- Go to: https://github.com/stamatak/standard-RAxML
- Click on the "Download ZIP" button on the right hand side. Move the downloaded folder into the 'RAXML' folder of BAGpipe.
- cd to this folder and rename the downloaded folder, in order to be able to use the standard path included in the pipeline. For example (*change the version name accordingly)*:

```
% cd BAGpipe/RAXML
% mv standard-RAxML-master standard-RAxML
% cd standard-RAxML
```

- Read the README file included in the downloaded folder, and compile the version that is more appropriate for your system, according to the instructions. For example, for the sequential SEE3 version:

```
% make -f Makefile.SSE3.gcc
% rm *.o
```

while for the pthreads SEE3 version:

```
% make -f Makefile.SSE3.PTHREADS.gcc
% rm *.o
```

Software Installation Mac OS X

1. **Xcode**

For compiling EMBOSS, RAxML and wget you will need to have Xcode installed.

- If you do not have Xcode installed, you can download it freely from https://developer.apple.com/xcode/ after registering.
- After installation, go to the "Downloads" tab in Xcode preferences and under "Components" push the "Install" button next to "Command Line Tools". Note that you need administration privileges for this installation.
- Make sure that the 'make' command works, by typing in a terminal window:

```
% make --help
```

2. wget

For the first steps of the pipeline (downloading sequence flatfiles and taxonomy from NCBI) you will need the command 'wget', which is not provided in Mac OS X. Note that you need administration privileges for this installation.

- Go to: http://ftp.gnu.org/pub/gnu/wget/ and download the latest version of wget in .tar.gz format.
- cd to Downloads and extract the downloaded file. For example (*adjust version number accordingly*):

```
% cd Downloads
% tar -xzf wget-1.14.tar.gz
```

- cd to the extracted folder and compile:

```
% cd wget-1.14
% ./configure --with-ssl=openssl
% make
% sudo make install
```

- Make sure that the command is working and then clean-up.

```
% wget --help
% cd ..
% rm -rf wget-*
```

3. NCBI BLAST

- Go to:
  http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download
  Or connect directly as 'Guest' to:
  ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
- Copy the appropriate zipped file for your system (for example, 'ncbi-blast-2.2.28+-universal-macosx.tar.gz') to the folder 'NCBI_BLAST' of the BAGpipe folder structure.

- cd to this folder and extract the downloaded zipped file. Rename the resulting folder in order to be able to use the standard path as included in the pipeline, and clean up if you want. For example (*you need to change the program version accordingly*):

```
% cd BAGpipe/NCBI_BLAST
% tar -zxvf ncbi-blast-2.2.28+-universal-
macosx.tar.gz
% mv ncbi-blast-2.2.28+ ncbi-blast
% rm ncbi-blast-2.2.28+-universal-macosx.tar.gz
```

4. USEARCH

- Go to: http://www.drive5.com/usearch/download.html
- Select the platform and the version you want to download. We currently recommend **v4.2.66**, as the pipeline has been built using this version. Newer versions will probably require some changes in the command line.
- Fill in your email address. You will immediately receive an email with a link for downloading the requested executable.
- Move the downloaded executable ('usearch4.2.66_i86osx32') to the 'USEARCH' folder in the folder structure of BAGpipe.
- Change file permissions in order to be able to execute it:

```
% cd BAGpipe/USEARCH
% chmod u+x usearch4.2.66_i86osx32
```

5. EMBOSS

- Go to: http://emboss.sourceforge.net/download/
  Or connect directly as 'Guest' to:
  ftp://emboss.open-bio.org/pub/EMBOSS/
- Copy the current version of the EMBOSS zipped file (for example 'EMBOSS-6.6.0.tar.gz') to the folder 'EMBOSS' of the BAGpipe folder structure.
- cd to this folder and extract the downloaded file. Rename the resulting folder (in order to be able to use the standard path included in the pipeline). Compile the program using configure and make. For example (*change the version number accordingly*):

```
% cd BAGpipe/EMBOSS
% tar -zxvf EMBOSS-6.6.0.tar.gz
% rm EMBOSS-6.6.0.tar.gz
% mv EMBOSS-6.6.0 EMBOSS
% cd EMBOSS
% ./configure
% make
```

6. MAFFT

- Go to: http://mafft.cbrc.jp/alignment/software/macportable.html
  and download the portable mac version of the program (for example, 'mafft-7.122-mac.zip'). The zipped file should be automatically unzipped.

- Move the downloaded folder 'mafft-mac' to the 'MAFFT' folder of the BAGpipe folder structure. Then cd to this folder, rename the downloaded folder ('mafft-mafft') which contains the executable mafft.bat.

```
% cd BAGpipe/MAFFT
% mv mafft-mac mafft
```

7. **RAxML**

- Go to: https://github.com/stamatak/standard-RAxML
- Click on the "Download zip" button. Move the downloaded folder into the 'RAXML' folder of BAGpipe.
- cd to this folder and rename the downloaded folder, in order to be able to use the standard path included in the pipeline. For example (*adjust the name of the version accordingly*):

```
% cd BAGpipe/RAXML
% mv standard-RAxML-master standard-RAxML
% cd standard-RAxML
```

- Read the README file included in the downloaded folder, and compile the version that is more appropriate for your system, according to the instructions. For example, for the sequential SEE3 version:

```
% make -f Makefile.SSE3.gcc
% rm *.o
```

BAGpipe steps

The procedure followed by BAGpipe is divided into two parts: 'Database construction' (steps 1.1 to 1.11) and 'Identification' (steps 2.1 to 2.15).

1. Database construction (see Figure 1)

The first part of the pipeline is designed to construct a *psbA-trnH* database by retrieving all the sequences homologous to this marker available in GenBank, and curating them so that they all have the same orientation and are trimmed to the length of this specific barcode region. With slight modifications it can be adapted to perform the same task for any other locus of interest. To make sure that all newly available GenBank sequences and latest updates are incorporated in the analyses, this procedure should be repeated every time that there is a new GenBank release, and consists of the following steps:

1.1. Retrieval of sequences and taxonomic information from the NCBI database

This is done by downloading the latest plant release (DNA) flatfiles from ftp://ftp.ncbi.nih.gov/genbank/ and the NCBI taxonomy database from ftp://ftp.ncbi.nih.gov/pub/taxonomy/

1.2. Generating taxon IDs for all Magnoliophyta (parse_ncbi_tax_database.pl)

The script *parse_ncbi_tax_database.pl* reads two files of the NCBI taxonomy database: 'nodes.dmp', which contains the tree-like structure of the taxonomic hierarchy, and 'names.dmp', which contains naming information for each taxonomic level, including synonyms, misspellings, etc. The script first reads 'nodes.dmp' and stores the hierarchy as a hash. It subsequently reads 'names.dmp' and stores the "scientific name" of each node. Next, starting from the user specified node, it traverses the hierarchy in a recursive manner. For each node, it builds a unique string based on the taxonomic names (from the start node) up to that node. The string is built by first discarding atypical characters (e.g., "(", ".", ":", ...) and taxonomic abbreviations (sp., aff., nr.), followed by taking the shortest unique substring of the current nodes appended to the string from parent nodes. A number is added when two rank nodes belonging to the same parent node start with the same letter. The script builds name strings to the species level, so lower taxonomic levels (e.g. subspecies) are disregarded. The script outputs a key, giving complete taxonomic information and NCBI taxon number, for each name string.

> EXAMPLE:
>
> A sequence linked to the following taxonomy: Magnoliophyta/ Eudicotyledons/ core_eudicotyledons/ rosids/ fabids/ Malpighiales/ Salicaceae/ Flacourtieae/ Xylosma_oligandra;
>
> would be automatically renamed as: Me1erfMSFXyoli

1.3. Create FASTA database from GenBank flatfiles (*create_fasta_database_from_genbank_flatfiles.pl*)

GenBank DNA data is released in flatfile formatted for different divisions. For example, the plant division (pln) is (currently) released in approximately 60 zipped files (64 zipped files in October 2013). Using the flatfile format is advantageous since it contains much additional information about the sequence entries, but it requires the relevant fields to be extracted. This script parses the data relevant for the pipeline purposes, and makes a single FASTA formatted file. The main fields parsed are: accession, NCBI taxon number, and DNA sequence, all of them used in the resulting FASTA file. Additional information (e.g., year, country of origin) is given in the log file. Further, and specifically related to our original implementation of the pipeline, the script looks for the string 'psbA-trnH' or 'trnH-psbA' in the gene and product fields of the flatfile, where the presence or absence of this string can be used for confirmation and protocol optimization purposes later if necessary. Model species with complete genome sequences or highly redundant sequence information in GenBank can be also purged out in this step for storage and computational reasons. The script is currently set to ignore the following species: *Arabidopsis thaliana, Glycine max, Medicago truncatula, Oryza sativa, Populus trichocarpa, Sorghum bicolor, Vitis vinifera* and *Zea mays*.

1.4. Assign taxon IDs to Magnoliophyta sequences (parse_taxon_from_fastafile.pl)

This script produces a FASTA file in which the taxon strings generated earlier are assigned to the FASTA database. Since taxon strings are made usually for a specific subset (i.e., Magnoliophyta), other sequences are thus removed here (notably, the fungi data is also contained in the pln division).

1.5. Search for sequences homologous to the psbA-trnH fragment (blastn)

This step screens the NCBI database for *psbA-trnH* by BLAST (*blastn*) searches using a representative and phylogenetically diverse set of angiosperm *psbA-trnH* query sequences, oriented in the same direction and covering the full length of the barcode region. In our implementation, a *blastn* search is performed using the standalone BLAST+ application and a query file including 967 representative *psbA-trnH* sequences, all provided in the same orientation, covering all Magnoliophyta orders and major families. Search parameter choices have been made according to the following reasoning: (1) Low complexity filter DUST is switched off since low complexity regions are abundant in the *psbA-trnH* marker. (2) Both strands are searched because there is no consensus for this marker in the orientation for sequence submission. (3) While many different e-values are reported for database searching, we found that using a relatively conserved value (i.e., 1E-6), we retrieved the great majority of the *psbA-trnH* annotated sequences. (4) Some default *blastn* parameters (typically num_descriptions & num_alignments, or max_target_seqs, depending on version used) had to be increased to obtain all homologous sequences available in GenBank. (5) The tabular output was chosen, because it is easier to parse than the standard BLAST output.

1.6. Retrieve homolog sequences, and reverse and complement them when required (parse_hits.pl)

This script is written to work on both BLAST and USEARCH (see below) tabular outputs produced with specific set of user fields (as selected by the relevant commands of the pipeline). In this step, the script parses the *blastn* output and prints the hits in full (without trimming). Not all the *psbA-trnH* sequences are submitted to GenBank in the same orientation (roughly, 65% in *psbA-trnH* direction and 35% in *trnH-psbA*), therefore the script automatically reverses and complements the database sequences that are in an opposite orientation compared to the queries. This is achieved by parsing the start and end positions of the BLAST output in relation to the queries.

1.7. Secondary similarity search with global alignment (*usearch_global*, *usearch_blastout_to_tabular.pl*)

An additional similarity search using USEARCH (Edgar, 2010) with global alignment and the same query file is performed against the BLAST hits retrieved in the previous step. The primary purpose of this step is to provide global alignments to be used for the subsequent trimming step (1.8), while it secondarily filters out some spurious hits (not truly homolog, for example some long chromosome sequences) retrieved by the BLAST search. Extensive testing has shown that trimming the sequences to the extent of the queries is much preferable when based on the *usearch_global* alignments rather than BLAST local alignments. The identity threshold is intentionally set to a very low value (0.1), to avoid losing some *psbA-trnH* sequences with very long indels. We use USEARCH version 4.2.66, because in the current version 6.0 there are memory limitations making it unsuitable for the size of typical datasets handled by BAGpipe. Unfortunately, version 4.2 contains a bug which affects the correct report of hit start/end positions in the tabular output, which is necessary for the trimming step. For this reason, we provide an additional custom script (*usearch_blastout_to_tabular.pl*) which converts the blast-output format of *usearch_global* to tabular format, required in the next step.

1.8. Retrieval of sequences, and trimming when necessary (parse_hits.pl)

This is the same script and procedure as described in step 1.6, but this time working with the USEARCH tabular output to trim the target sequences when necessary, according to their start and end positions in relation to the query. As many queries are used, a given target sequence may be present with a number of different start/end positions. Two trimming options are implemented, trimming according to the longest hit, or trimming according to the left-most and right-most positions over all hits. The latter was found to be problematic for *psbA-trnH* due to some long genomic sequences that remained untrimmed. Thus, for our original use of the pipeline, the former is the default option.

1.9. Filter out redundant sequences of the same species (dereplicate.pl)

This step is considered necessary in order to remove from the reference database the excess of identical and nearly identical sequences that exist in GenBank for some species that have been studied extensively at the population level. For this purpose all sequences belonging to a given species are first retrieved, then a *blastn* all-against-all search is performed with the same settings as described previously. Using the BLAST percent identity values, single linkage clustering is performed using a separate script (*single_linkage_1.02.pl*), followed by random removal of sequences showing similarity above the specified similarity threshold (our default value is 99.8%). This process is iterated over all species.

1.10. Removal of sequences with wrong annotations

Errors in the taxonomic annotations of GenBank sequences will affect the following identification steps of the pipeline. At this stage, we automatically remove all GenBank accession numbers that we suspect to have wrong taxonomic annotations. A tentative list of these accession numbers that we recognized as erroneous is already provided within the pipeline, but this is not exhaustive and will need to be continuously updated.

1.11. Optional addition of locally available sequences to blastable database

Any other locally available sequences that have not been released in GenBank yet can be added manually to the database at this stage.

2. Sequence identification (see Figure 2)

In the second part of the pipeline, the user provides a set of query sequences in a FASTA formatted file (using the same sequence orientation as provided in step

1.5) and these are processed for taxonomic assignment. In order to maximise the efficiency of the process and reduce computational demands of subsequent steps, the queries are initially clustered into groups of similar sequences (steps 2.1 and 2.2). Each of the resulting ‘query-groups’ (which very roughly correspond to representatives of the same plant family) is processed iteratively using a loop structure. The objective of this clustering procedure is to speed up and facilitate multiple sequence alignment (alignment problems are often encountered in the *psbA-trnH* marker) (steps 2.8 and 2.9), but also phylogenetic inference steps (steps 2.11-2.13), especially in a multi-core system where all these processes can be parallelised. The sequence identification pipeline consists of the following steps:

2.1. Similarity search among query sequences (usearch_global)

Similarity searches among the query sequences (all-against-all) are performed using the *usearch* algorithm (USEARCH version 4.2.66; Edgar, 2010) with global alignment and a 0.85 default identity threshold. The USEARCH identity scores are used in the next step for sequence clustering.

2.2. Clustering of query sequences (single_linkage_1.02.pl)

Query sequences are clustered into ‘query-groups’ at the 0.85 identity level. This is achieved by means of the average neighbour linkage and the *usearch_global* identity scores from the previous step. The following steps are performed iteratively for each of the retrieved ‘query-groups’ and using a loop structure.

2.3. Similarity search against the reference database (blast_query_clusters.pl)

This step performs similarity searches for each of the query sequences in a cluster (='query-group') and against the custom *psbA-trnH* database created in step 1.11. In brief, the script obtains all sequences from each query-group, uses them individually as queries against the *psbA-trnH* database, and retrieves their respective homolog sequences. The script provides several options to conduct the homology search, including BLAST, USEARCH v.4 and USEARCH v.6. The default and recommended option is USEARCH v.4 and using a 0.8 identity threshold.

2.4. Estimation of p-distances between query sequences and retrieved homologs (calculate_pairwise_distances.pl)

In this step, pairwise p-distances are calculated between each query sequence in a query-group and all the homologs retrieved for this group. Considering that length variation is very common for the default marker used in BAGpipe (*psbA-trnH*), distances are estimated taking gaps into account, but counting each string of gaps as a single event in order to avoid inflating the obtained values. These distances are obtained for each pair of sequences aligned using the Needleman-Wunsch pairwise sequence alignment algorithm as implemented in the *needle* tool of the EMBOSS v. 6.5.7 package (Rice et al., 2000), under gap opening penalty of 10 and gap extension penalty of 0.5. Sequence similarity between aligned pairs is scored by the provided custom script considering (a) each gap as a single state change, (b) terminal gaps ignored, and (c) ambiguous characters (e.g., NYRMWSKVHDB) ignored.

2.5. Output distance results (process_distance_results.pl)

The results of distance analysis are processed and summarized, producing the following two outputs:

- Text file summarizing the p-distance results, which includes for each query sequence: (a) database sequence(s) producing the 'best match'; and (b) the common part of the taxonomy IDs in all database sequences which are within a 1% and 4% divergence threshold relative to each query sequence (these thresholds were determined analytically as meaningful for *psbA-trnH* identifications in Papadopoulou et al., submitted).
- Text file with a list of all database sequences within 0<p-distance<0.10 from each query sequence, ranked by increasing distance, including full species name and taxonomy, as well as sequence length information and length of pairwise alignment between the query and the database sequence.

2.6. Retrieval of sequences below a certain p-distance threshold (parse_hits.pl)

The same script as used in steps 1.6 and 1.8 is employed here again for sequence retrieval. In this case, homolog sequences retrieved in step 2.3 are filtered based on the calculated p-distances, and using a 10% divergence threshold as default (in practice this needs to be given as a 0.9 similarity threshold, instead of a divergence value, since this script works with similarity scores).

2.7. Assessment of data size: tallying number of retrieved sequences per query-group

While in principle there is no limitation for the size of datasets to be analyzed using phylogenetic procedure, for the sake of speed and efficiency of the process, it is recommended to restrict the analyses to manageable data size. Thus, it is advisable to stop the pipeline in this step and check the number of retrieved sequences per query-group. The goal would be to have more or less size-normalized datasets in the range of 10-500 sequences each. To achieve this, the distance-threshold of the previous step (2.6) can be adjusted increasing it to augment database sequence retrieval for very small datasets or decreasing it to reduce the size of very big datasets.

2.8. Multiple sequence alignment (mafft E-INS-i)

In this stage, data compiled in the previous steps, i.e. the queries plus their respective distance-filtered homolog sequences, are aligned using MAFFT v7.043b (Katoh et al., 2002; Katoh & Standley, 2013) with the E-INS-i algorithm (other options are available, but this proved the better trade-off between alignment accuracy and speed; Papadopoulou et al., submitted). Depending on the size of each dataset, this step may take long to run; thus, if a multi-core processor is available, it is advisable to run several jobs in parallel (but obviously limiting the number of concurrent processes to the number of available cores).

2.9. Indel recoding and matrix preparation (2xread.pl, concatenate_v2.pl)

Indels in the resulting alignment are recoded as binary characters using 'Simple indel coding' (Simmons & Ochoterena, 2000) as implemented in the '*2xread.pl*' script (Little, 2005). As a result of this step, the pipeline produces a matrix including both DNA and binary character data, as well as a partition file for RAxML (*concatenate_v2.pl).*

2.10. Conversion of FASTA to phylip format for RAxML (format_conversion.pl)

This script simply converts the matrix file from FASTA to relaxed phylip format (i.e., allowing for taxon names to be longer than ten characters).

2.11. Tree inference (RAxML)

The matrix produced in the previous steps is analyzed with RAxML (Stamatakis, 2006) using a mixed model for binary+DNA data. Tree search is based on 20 independent searches, starting from random stepwise addition parsimony trees. Moreover, clade support is assessed by rapid bootstrapping (Stamatakis et al., 2008) with 100 pseudoreplicates. As with the alignment, depending on the size of each dataset, this step may slow down the process significantly. So it is important to use the best RAxML version for your system (see RAxML README file) and explore the possibilities for parallelisation if a multi-core processor is available.

2.12. Tree rerooting (midpoint_root.pl, gjonewicklib.pm)

Taxonomic assignment based on a tree topology and clade support navigates the tree from unassigned terminal nodes towards supported inner nodes and parses the taxonomy of identified members of this clade to extrapolate the subtending taxonomy. For this reason, it is critical to work with a tree with correct polarization. The outgroup method is not available in our implementation of BAGpipe given the dynamic nature of database assemblage for each inference. Alternatively, we empirically recognized (by contrasting dozens of obtained trees with the current knowledge of plant systematics) that midpoint rooting is the best polarization strategy for our automated procedure. Thus, the best ML tree obtained by RAxML is rerooted using the midpoint strategy as allowed by the 'gjonewicklib' library, part of the SEED toolkit (http://www.theseed.org/).

2.13. Adding support to the midpoint-rooted tree (CompareToBootstrap.pl, MOTree.pm)

Bootstrap values obtained after the RAxML analysis are added to the rerooted tree files using *CompareToBootstrap.pl* (by Morgan Rice, http://www.microbesonline.org/fasttree/treecmp.html).

2.14. Parsing clades for taxonomic identification of queries

In this step, the clades to which each query sequence belongs are parsed from the resulting tree files. The part of the taxonomy (e.g., unique text string from step 1.2) shared by all database sequences belonging to this clade is used to assign the taxonomy of the queries. Only supported clades (default: ≥70%) are considered in this step. A text file is generated including information of clade membership for each query sequence. Two tree-based taxonomic assignment strategies are implemented in BAGpipe: 'outer clade' ('strict' criterion: two consecutive supported nodes are considered), and 'inner clade' ('liberal' criterion: supported sister group relationships are considered). The following information is given for both strategies: (a) bootstrap support value of the clade used for the assignment, (b) taxonomic ID of the clade (common part of the taxonomy of all database sequences included in the clade), (c) names of all database sequences included in the clade, and (d) names of all other query sequences included in the clade.

2.15 Formatting of taxon labels (format_newick_IDs.pl)

This script reverses the Taxon ID of step 1.2 to produce full species names. Tree labels of the midpoint-rooted trees with bootstrap values are thus reformatted to include full species names, for a more user-friendly visualization.

Running BAGpipe

As explained before, BAGpipe is divided into two parts. The first part ('pipeline1_database', steps 1.1 to 1.11) should be run when there is a new GenBank release (approximately once every 2 months), while the second part ('pipeline2_identification', steps 2.1 to 2.15) applies when there are new query sequences to identify. Some general suggestions are applicable to both parts:

- Each part can be run in an automatic or semi-automatic mode, but we strongly recommend that the **first time** the pipeline is used **each command is run separately** (by copying and pasting in terminal) to make sure that the whole process runs smoothly before trying the automatic mode and identify possible problems or changes required to scripts or their parameters beforehand.
- Each step in the pipeline can be used independently of the others by commenting out the other steps using #
- Every time that the pipeline is run, the files generated in the previous run will be **over-written**. If these previous files are to be kept, they should be moved to another folder beforehand. For example:

```
% mkdir MyFirstDatabase
% mv BAGpipe/Database/all_psbAtrnH* MyFirstDatabase
% mv BAGpipe/Database/key* MyFirstDatabase
```

  or

```
% mkdir MyFirstResults
% mv BAGpipe/Identification/*query_group*
MyFirstResults
```

- If jobs are sent **remotely** to another computer, 'nohup' should be used before executing the pipeline, so that the terminal window can be exited without stopping the pipeline from running. For example:

```
% nohup ./pipeline1_database &
```

Pipeline1_database

In order to start running the database construction step, cd to the 'BAGpipe/Database/' folder.

```
% cd BAGpipe/Database
```

There are two files in this folder. The file **'queries_psbAtrnH.fas'** is a FASTA file of 967 representative *psbA-trnH* sequences to be used as queries for the construction of the *psbA-trnH* database (it helps to retrieve all homologous sequences from GenBank and allows trimming them to match the size of the selected fragment). The example query sequences are all oriented in the same direction, and they cover all Angiosperm orders and additionally all major taxa known to occur in Nicaragua (our original geographic scope). Since this query file has been produced specifically for a project on the Nicaraguan SDTF flora (see Papadopoulou et al., submitted) it might not be optimal for other projects. This file can be substituted by an equivalent one with a different set of query sequences, in which case it can either be given the same file name, or a different one, but changing the name accordingly in the command lines for steps 1.5 and 1.7 of the pipeline.

The text file **'pipeline1_database'** contains all the pipeline commands and relevant comments and instructions. To customize it, it is only required to open the file in any text editor. All paths assume that the commands for every step are called from within the 'BAGpipe/Database/' folder (it is strongly advised to preserve this structure). On the top of each command there is a short description of what the command does, the default names of input and output files, any other programs that the command depends on, as well as the most common options that the user may want to change.

Indeed, users may need to change some of the options before running them. If the pipeline is used specifically for **Angiosperm *psbA-trnH*** a good starting point is revising the following points (1) and (2). Users familiar with the pipeline steps can make additional changes, but in principle the default parameters of the pipeline have been specifically optimised for this marker. Users interested in a **different marker or taxon** should revise all the points listed in (3) to (5), and make any relevant changes in order to optimise the settings for the specific dataset.

Most common changes and points to remember:

(1) The user may want to use a **custom query file**, in which case the example file BAGpipe/Database/queries_psbAtrnH.fas has to be substituted OR alternatively a different file must be placed in the 'BAGpipe/Database/' folder and the name of the query file has to be changed accordingly in steps 1.5 and 1.7 of the pipeline. These changes are obviously necessary when using BAGpipe for a **different marker** (not

psbA-trnH). **IMPORTANT:** the sequences of the query file must be provided all in the **same orientation**.

(2) In order to add **locally generated sequences** (i.e., not submitted to GenBank) to the database in step 1.11:

- These need to be in the same format as the other database sequences, i.e. with a label containing TaxonID_YourSequenceID, For example a sequence of the species *Xylosma oligandra* could be labelled as Me1erfMSFXyoli_<sequence_ID>. Taxon ID codes can be found in the keyfile 'key_Magnoliophyta' either using the species name (Xylosma_oligandra), or the NCBI taxon number (681507). The latter might be advantageous in cases of synonymy, i.e. when a custom species name does not coincide with the NCBI taxonomy.
- If a species is not included in the keyfile, i.e. if it is a new species for GenBank, then a new code has to be made for it. This can be done using the code of the genus (or of the next higher taxonomic rank in GenBank) and creating a unique code (e.g. 'Me1erfMSFXyNew1' could be used for *Xylosma characantha*, since 'Me1erfMSFXy' is already the code for the genus *Xylosma*). A corresponding line has to be added in the keyfile, copying the format of the existing lines, e.g.:

  MeeurfMSFXyNew1 Xylosma_characantha 0000 species no_rank:eudicotyledons no_rank:eudicotyledons subclass:rosids no_rank:fabids order:Malpighiales family:Salicaceae tribe:Flacourtieae genus:Xylosma species:characantha

- **Advice:** If there are many taxa to be added, the Linux/Unix 'grep' command can be used to retrieve taxon IDs and taxonomy strings from the keyfile in a batch mode. Additionally, the *get_taxonomy.py* Python script (by Robert Lanfear) can be used to help with getting the NCBI taxon numbers for a list of taxa and directly from GenBank (please read notes within the script and README file for instructions). This can help to avoid synonymy problems (e.g., in the 'taxonomy.txt' example file included in the 'get_taxonomy' folder, the taxon 'Stizolobium pruriens' is identified as synonym of 'Mucuna pruriens'). The original 'get_taxonomy.py' script works with species names (i.e., Linnean binomials), but if NCBI taxon numbers for higher taxonomic ranks (genus or higher) are preferred, then it is possible to use the modified version of the script: *get_taxonomyHigherRank.py*. These scripts are available in our rendering of BAGpipe in the 'BAGpipe/z_additional_scripts/' folder.

(3) For users interested in a marker other than *psbA-trnH*, the following changes may be considered:

- Step 1.3: change the annotation string within the script *create_fasta_database_from_genbank_flatfiles.pl*. This change is not strictly necessary for the pipeline to work correctly; it is only for confirmation purposes and protocol optimization (see point 7 below regarding the additional script *append_name_matches.pl*).
- Step 1.5: E-value cut-off for sequence retrieval (default: 1E-6).

- Steps 1.6 and 1.8: length cut-off for retaining a sequence (default: 200 nt).
- Step 1.9: filter threshold used to remove nearly identical sequences (default: 99.8).
- Step 1.10: update of accession numbers to be removed from the analysis because they correspond to sequences with wrong taxonomic annotation (these may be detected *a posteriori* from user analyses and it is useful to keep track of them in this step to avoid noise in subsequent BAGpipe runs).

(4) For users interested in a **different taxon** (not angiosperms), changes above apply, and additionally:

- Step 1.1: change the name of the flatfiles that are downloaded from GenBank.
- Step 1.2: change the NCBI_taxonomy_ID (3398 = Magnoliophyta).
- Step 1.3 step: change the name of the flatfiles within the *create_fasta_database_from_genbank_flatfiles.pl* script (***NOTE:** This is not done in the command line, but changes must be done in the *create_fasta_database_from_genbank_flatfiles.pl* script itself, found in the folder 'Scripts1_database').

(5) In order to **optimise** the pipeline for a different marker and/or taxon, it might be worth using the *append_name_matches.pl* script (found in the 'z_additional_scripts' folder). This script can be used to compare the sequences that are retrieved by the similarity searches in steps 1.5 and 1.7 with the sequences that would be retrieved using gene annotation. See instructions for use within the 'append_name_matches' folder.

Running BAGpipe database steps in automatic mode

Once the optimization of individual commands is satisfactory, the 'pipeline1_database' can be run in an automatic mode, and after changing file permissions using 'chmod':

```
% chmod u+x ./pipeline1_database
% ./pipeline1_database
```

Pipeline2_identification

Once the reference database has been assembled successfully (the necessary files for subsequent steps are: '*all_psbAtrnH.filtered**' and '*key_Magnoliophyta*'), it is possible to start the sequence identification procedure.

cd to the 'BAGpipe/Identification/' folder.

```
% cd BAGpipe/Identification
```

There are four files in this folder. The text file **'pipeline2_identification'** contains all the pipeline commands, and relevant comments and instructions. As before, **it is not recommended to run the whole 'pipeline2_identification' part for the first time in an automatic mode**. It is strongly recommended to go through each step separately, making any necessary changes and **running each command separately** (by copying and pasting in your terminal) in order to be able to identify easier any problems (see points 3 and 4 below for further details). All paths assume that commands are called from within the 'BAGpipe/Identification/' folder. Custom changes to the 'pipeline2_identification' file can be done using any text editor. On the top of each command there is a short description of what it does, the default names of input and output files, any other programs that the command depends on and the most common options that could be changed to optimize its performance.

Most common changes and points to remember:

> (1) **Input file:** The file '**queriesSeqs.txt**' is an example query file of 21 herbivore diet sequences. It can be used to make a test run and confirm that the pipeline works fine on the user's system. This file should be substituted with a user FASTA file of query sequences, but keeping the same file name. It is critical that all query sequences in the file are in the same orientation, the same used in the reference database (as in the 'queries_psbAtrnH.fas' of step 1.5), and that sequence names do not contain any spaces, either within the name or after (*NOTE: if the FASTA file is exported directly from Geneious, the software adds a blank space at the end of the sequence name, which will cause problems in step 2.2, so it is important to remove it).
>
> (2) Steps 2.4 and onwards are performed iteratively for each query-group using a loop structure. For this purpose, the number of groups is counted in step 2.3 and given a variable name, which is used in subsequent steps to form the loop. However, **if the pipeline stops running in this step** the name of the variable will not be saved. In order to make it possible to salvage previous results and continue running later the procedure from the same point where it stopped, **step 2.3 (the second part) has to be repeated** to name the variable again. Alternatively, once the number of query groups is known, **the command line can be edited** accordingly (e.g., find and replace "`for i in %(seq 0 %number)`" with "`for i in {0..99}`" in pipeline2_identification (or pipeline2a_distances).
>
> (3) For increased efficiency, especially when there are time or computational resource limitations, we suggest to stop the procedure after performing steps 2.1 to 2.7 (i.e. all steps included in pipeline2a_distances) and **checking the number of retrieved sequences per query-group** (as printed out in step 2.7). The number of the very big (>500 sequences) or very small datasets (<10 sequences) can be tuned by altering the applied p-distance threshold for specific query-groups.
>
> To reduce the number of retrieved homologs, step 2.6 can be repeated with a lower p-distance threshold (<0.1). As the script *parse_hits.pl* works on similarity scores rather than distances, this threshold has to be given as an increase in similarity score (e.g., >0.9). For example, in order to reduce the

number of retrieved sequences for query-groups 1, 35 and 115, one could run:

```
% for i in 1 35 115; do echo "query group %i NW
alignment";
perl ../Scripts1_database/parse_hits.pl
query_group.%i.homologs
query_group.%i.fas.needle_distances 0.92 0 200
done
```

On the contrary, to increase the number of retrieved homologs, step 2.6 has to be repeated with a higher p-distance threshold (>0.1), i.e. a lower similarity score. In the specific case of the *psbA-trnH* marker, we do not recommend reducing the similarity score below 0.85 (i.e., equivalent of 0.15 p-distance), because this level of divergence often results problematic for multiple sequence alignment.

(4) As we consider very important to keep the number of retrieved homologs in a manageable size for efficient multiple alignment and phylogeny steps, we provide the pipeline in two separate files: **'pipeline2a_distances'** (steps 2.1 to 2.7) and **'pipeline2b_phylogeny'** (steps 2.8 to 2.15). Both can be run in automatic mode, but their split in two processes facilitates an intermediate "manual" controlling step. Note that in the 'pipeline2b_phylogeny' part, by default the user needs to set the number of groups for the loop structures, by finding and replacing the "`for i in {0..999}`", with the correct query-group number. This number has to consider the **total number of groups**, including cases that have not retrieved any homologs from the database.

(5) The multiple alignment (MAFFT in step 2.8) and phylogenetic inference (RAxML in 2.11) steps should be **parallelised if a multi-core processor is available** (but limiting the number of concurrent processes to the number of available cores). The command 2.8.3 (line 36 of the pipeline2b_phylogeny file) will count the number of available CPUs on the system and the jobs will be distributed accordingly in the next steps (2.8.3 and 2.11). These commands should work fine with MAFFT and with the sequential version of RAxML. However, they will not work with the PTHREADS version of RAxML. If this version is used, it is recommended to install and use the **'parallel' command** from the package **'moreutils'.** On many Linux systems it can be installed automatically. To check if it is installed already and, in case it is not, to be questioned about whether to install it or not:

```
% parallel –help
Command not found. Install package 'moreutils' to
provide command 'parallel'? [N/y]
```

When it is installed, it can be used with the PHTREADS version of RAxML as follows (after adjusting the number of query groups in `{0..999}`):

```
% parallel -i ../RAXML/standard-RAxML/raxmlHPC-
PTHREADS-SSE3 -T 2 -s
query_group.{}.all_characters.phy -n query_group.{}
-m GTRCAT -c 4 -f a -x 12345 -p 12345 -# 100 -q
query_group.{}.partitionfile -- {0..999}
```

Another alternative towards the same end is the **GNU-parallel**, available at: http://www.gnu.org/software/parallel/. Note that there is a name collision between the two 'parallel' commands, so in order to keep things simple, it is better to choose and install only one of them.

(6) If a **different marker or taxon** are used, the following settings may be changed:

- Steps 2.1 and 2.2: similarity thresholds in clustering step (default: 85%).
- Step 2.3: similarity threshold to retrieve sequences from database with *usearch_global* (default: 80%).
- Step 2.6: p-distance threshold to filter retrieved sequences (default: 0.9 similarity = 0.1 distance).
- Step 2.8: multiple alignment algorithm (default: MAFFT E-INS-i)
- Step 2.9: indel recoding (default: SIC).

Running BAGpipe identification steps in automatic mode

Once individual commands are optimised and selected for the specific analytical needs, '**pipeline2a_distances**' can be run in an automatic mode, after changing file permissions using 'chmod':

```
% chmod u+x ./pipeline2a_distances
% ./pipeline2a_distances
```

The final step (2.7) outputs the number of retrieved sequences per query group. We suggest that you look at these numbers and adjust them as explained in point (3) above. Then set the correct number of query-groups in 'pipeline2b_phylogeny', consider the parallelisation issue and run it, after changing permissions:

```
% chmod u+x ./pipeline2b_phylogeny
% ./pipeline2b_phylogeny
```

Alternatively, if you do not want to adjust the number of retrieved sequences in step 2.7, you can run the whole 'pipeline2_identification' (steps 2.1 to 2.15) at one go (NOT recommended):

```
% chmod u+x ./pipeline2_identification
% ./pipeline2_identification
```

Main results files

The following are the main results of BAGpipe presented in four different files (see Table 1 for details about the column headings of each file):

(1) ALL_DISTANCE_RESULTS.0.1.txt: Text file listing all database sequences within 0<p-distance<0.10 from each query sequence, ranked according to increasing distance, including full species name and taxonomy, as well as sequence length information and length of pairwise alignment.
(2) ALL_DISTANCE_RESULTS_SUMMARY.0.1.txt: Text file summarising the p-distance results, which includes for each query sequence: (a) database sequence(s) providing the 'best match', and (b) the common part of the taxonomy of all database sequences both within a 1% and a 4% threshold.
(3) ALL_TREE_BASED_RESULTS.txt: Text file including information of clade membership for each query sequence. The following information is given for both 'outer clade' ('strict' criterion), and 'inner clade' ('liberal' criterion) (see comments to step 2.14, p. 14): (a) bootstrap support value for the clade, (b) taxon ID of the clade (i.e., common part of the taxonomy of all database sequences included in the clade), (c) names of all database sequences included in the clade, and (d) names of all other query sequences included in the clade.
(4) RAxML_bestTree.query_group.$i.rerooted.bootstrap.reformatted: Midpoint-rooted RAxML best trees with bootstrap values, and reformatted labels to include full species names.

(*NOTE: Some query sequences may not appear in all or some of the result files. This will happen when they do not retrieve enough homologs from the database given the selected thresholds.)

Table 1. Explanation of column headings in the main results files.

ALL_DISTANCE_RESULTS	
query_ID	query sequence label
hit_ID	database sequence label
p_distance	p-distance between query and database sequence
alignment_length_with_gaps	total length of pairwise alignment between query and database sequence
alignment_length_without_gaps	length of pairwise alingment without positions with gaps
length_of_query_sequence	length of query sequence
length_of_target	length of database sequence
species_label	species to which the database sequence belongs
lineage	taxonomy of database sequence
ALL_DISTANCE_RESULTS_SUMMARY	
queryID	query sequence label
best_match_(distance)	database sequence with minimum p-distance from query (p-distance value)
shared_taxon_at_0.99	common part of the taxonomy of all database sequences within a 1% p-distance from the query
shared_taxon_at_0.96	common part of the taxonomy of all database sequences within a 4% p-distance from the query
ALL_TREE_BASED_RESULTS	
query_member	query sequence label
outer_support	bootstrap support of the 'outer' clade, i.e. following a 'strict' criterion
outer_taxon_name	common part of the taxonomy of all database sequences belonging to the 'outer' clade
outer_species_list	all database species belonging to the 'outer' clade
outer_nonqueries	all database sequences belonging to the 'outer' clade
outer_queries	other query sequences belonging to the 'outer' clade
inner_support	bootstrap support of the 'inner' clade, i.e. following a 'liberal' criterion
inner_taxon_name	common part of the taxonomy of all database sequences belonging to the 'inner' clade
inner_species_list	all database species belonging to the 'inner' clade
inner_nonqueries	all database sequences belonging to the 'inner' clade
inner_queries	other query sequences belonging to the 'inner' clade

Credits, contact us and citation

LICENSE INFORMATION:

CONTACTS:

If you have any problems or questions related to BAGpipe, please contact us:

Anna Papadopoulou: a.papadopoulou05@alumni.imperial.ac.uk

Douglas Chesters: dc0357548934@live.co.uk

Jesús Gómez-Zurita: j.gomez-zurita@ibe.upf-csic.es

CITATION:

If you use BAGpipe for your research please cite:

Papadopoulou A, Chesters D, Coronado I, De la Cadena G, Cardoso A, Reyes JC, Maes J-M, Rueda RM & Gómez-Zurita J. (2014) Automated DNA-based plant identification for large-scale biodiversity assessment. *Mol. Ecol. Res. (In press)*