Cladistics

Cladistics (2012) 1–11

10.1111/j.1096-0031.2012.00400.x

GB-to-TNT: facilitating creation of matrices from GenBank and diagnosis of results in TNT

Pablo A. Goloboff[a,b] and Santiago A. Catalano[a,c]

[a]*Consejo Nacional de Investigaciones Científicas y Técnicas, Miguel Lillo 205, 4000 S.M. de Tucumán, Argentina;* [b]*Instituto Miguel Lillo, Facultad de Ciencias Naturales, Miguel Lillo 205, 4000 S.M. de Tucumán, Argentina;* [c]*Fundación Miguel Lillo, Miguel Lillo 251, 4000 S.M. de Tucumán, Argentina*

Accepted 12 March 2012

Abstract

This paper presents a pipeline, implemented in an open-source program called **GB→TNT** (GenBank-to-TNT), for creating large molecular matrices, starting from GenBank files and finishing with **TNT** matrices which incorporate taxonomic information in the terminal names. **GB→TNT** is designed to retrieve a defined genomic region from a bulk of sequences included in a GenBank file. The user defines the genomic region to be retrieved and several filters (genome, length of the sequence, taxonomic group, etc.); each genomic region represents a different data block in the final **TNT** matrix. **GB→TNT** first generates Fasta files from the input GenBank files, then creates an alignment for each of those (by calling an alignment program), and finally merges all the aligned files into a single **TNT** matrix. The new version of **TNT** can make use of the taxonomic information contained in the terminal names, allowing easy diagnosis of results, evaluation of fit between the trees and the taxonomy, and automatic labelling or colouring of tree branches with the taxonomic groups they represent.

A huge proportion of published phylogenetic studies use molecular sequences as a source of data for phylogenetic analyses. The numbers of both genes and species included in those analyses increases yearly, and this trend will no doubt accelerate with next-generation sequencing. It has been shown (e.g. Goloboff et al., 2009; Smith et al., 2011) that algorithms for phylogenetic calculations, once considered the main limitation for large analyses, can handle matrices of sizes far beyond what most users need. In fact, with these algorithmic advances, the most serious problem most users face are of a different nature, among them: (i) how to easily generate a matrix when several genomic regions are included, and (ii) how to evaluate and visualize the results of large trees. These two problems can be overcome by means of semi-automatic procedures. Here we present **GB→TNT** (GenBank-to-TNT), an open source program which implements an easy-to-use pipeline for the creation and analyses of large molecular matrices, which starts from GenBank (GB) files and finishes with **TNT** matrices. The **TNT** matrices incorporate all the taxonomic information for each terminal taxon, which the new version of **TNT** can use for easily diagnosing the results, automatically labelling tree branches with the taxonomic groups they represent, or colouring different taxonomic groups.

*Corresponding author:
E-mail address: pablogolo@csnat.unt.edu.ar

Other contributions

For matrix creation, the program most similar to **GB→TNT** is **SequenceMatrix** (Vaidya et al., 2010), which can be used to create **TNT** matrices by merging aligned Fasta files into **TNT**-format data; the program allows for combination of different genes and taxon sets, as well as other facilities. The main differences with the program presented here are that **SequenceMatrix** requires that the data have already been retrieved from GenBank and aligned, that it does not facilitate incorporating taxonomic information in the taxon names, and that it does not allow for checking of spelling errors in taxon names. As **SequenceMatrix** implements some facilities not available in **GB→TNT**, one could well export the Fasta aligned files produced by **GB→TNT** (with taxonomic information already incorporated in the taxon names) into **SequenceMatrix**, to obtain the benefits of both programs.

For tree-visualization, there are a large number of programs available, for example **TreeView** (Page, 1996), **Paloverde** (Sanderson, 2006), **FigTree** (Rambaut, 2009), **ScriptTree** (Chevenet et al., 2010), **TreeGraph** (Stöver and Müller, 2010), and **Dendroscope** (Huson et al., 2011). Several of those programs have useful and intuitive facilities for colouring tree branches, and a nice graphical output. However, semi-automatic coloration and branch labelling are not easily compatible with **TNT**-format files, and facilities oriented towards testing or displaying taxonomic groups are either entirely absent or hard to automate (the colour or labelling has to be defined on a tree-by-tree and branch-by-branch basis, so that plotting large trees with taxonomic information becomes impractical). In addition, the only one of those programs that could plot really large trees (e.g. Goloboff et al., 2009 tree of 73 060 taxa) is **Dendroscope**; all the other programs crashed or froze. Thus, we have chosen to improve the abilities for tree-displaying in **TNT**, to allow integrating the analysis and diagnosis of taxonomic results into a single program, which works efficiently for large data sets and trees. The main difference between the new implementation and existing tree-plotting programs is that, in **TNT**, the taxonomy is defined as a reference external to the tree(s), and any tree can be easily checked against it.

Overview of GB→TNT

GB→TNT is a Windows open-source program for easily creating data sets for **TNT** (Goloboff et al., 2003, 2008) from GB files. It is a merge of the **C** scripts (gb2tnt and fas2fas, used by Goloboff et al., 2009, in their analysis of 73 060 eukaryotes), with a graphic interface and several functionalities added. It is distributed together with all Windows versions of TNT, and can also be downloaded separately from (http://www.zmuc.dk/public/phylogeny/TNT/GenBank-to-TNT.zip). The source code (as of this writing, ca. 7500 lines of code, licensed under GPL ver. 3) is included in the package; we have compiled it with **OpenWatcom** (a free, open-source **C** compiler, available at http://www.openwatcom.org).

The input for the program is sequence data in GB format, and the final output is a **TNT** matrix in which each block represents a different genomic region (Fig. 1). In addition to GB files, Fasta, aligned Fasta, and TNT files can also be used as input for **GB→TNT**.

GB→TNT is designed to retrieve a defined genomic region from a bulk of sequences included in a GB file.

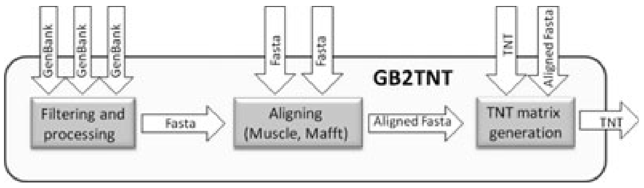


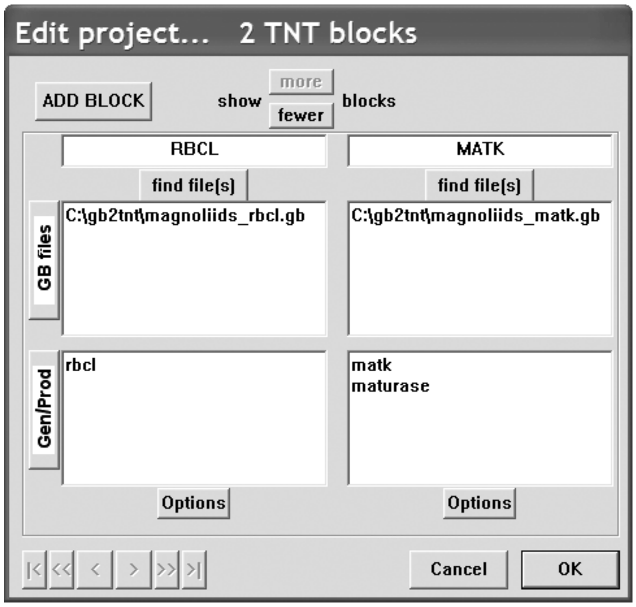
Fig. 1. A flowchart showing the steps involved in the **GB→TNT** pipeline. Arrows represent the format of input and output files.

The user defines the name of the genomic region (usually a gene) to be retrieved and also where the program will look for that name in the GB file(s). In addition, the user can define several filters: genome, sequence length, taxonomy, etc. Each genomic region to be parsed is included in a different block of the project that will also constitute a different block in the final **TNT** matrix.

GB→TNT first generates Fasta files from the input GB files, optionally retaining (see below) either one (in this case, the longest) or multiple sequences per species. Then, an alignment is generated from each Fasta file by calling an alignment program—**Mafft** (Katoh, 2008), **Muscle** (Edgar, 2004), or any alternative program defined by the user. Finally, all the aligned files are merged into a single **TNT** matrix.

GB→TNT projects

All the information and settings to parse the GB files is organized in a **GB→TNT** *project*. A *project* includes the definition of the **TNT** data blocks, as well as their input and output settings. Each block of a **GB→TNT** project will correspond, in principle, to a different genomic region to be parsed. The general options are set with the **Edit Project** dialog (**File/Edit Project**). This dialog presents two boxes for each block (Fig. 2); in the top one the user defines the location of the input files to be parsed, while in the lower one the user defines the name(s) of the genomic region to be retrieved. A project can also include blocks where the input files are Fasta (*fas* extension), aligned Fasta (*aln* extension), and **TNT** (native **TNT** files, *tnt* extension); evidently, for those types of input file, there is no need to fill the bottom box with a product or gene name. **GB→TNT** uses the different types of input file at different points of the pipeline (Fig. 1); for example, the alignment step is skipped for files of types *aln* and *tnt*. Note that the criteria for inclusion or exclusion that are defined in the **Taxonomy** menu, as well as the filter for sequence length defined in the **Block Options** dialog (subdialog of **Edit Project**), apply to GB and Fasta files but not to aligned Fasta and **TNT** files—if the user wants such filters, files of these types must have been processed ahead of time.


Edit project... 2 TNT blocks

RBCL

GB files: C:\gb2tnt\magnoliids_rbcl.gb

Gen/Prod: rbcl

MATK

GB files: C:\gb2tnt\magnoliids_matk.gb

Gen/Prod: matk, maturase

Fig. 2. "Edit project" dialog; this is used to define the main organization of the data set, including input files and genes to retrieve from them.

"Block Options" dialog

The specific settings for each block (Fig. 3) are defined by clicking the "**Options**" button below the bottom ("**Gen/Prod**") box. To save work, in the case of settings common to many blocks, it is possible to define them for one block, and then copy them onto a set of specified blocks, with the **Copy Options to Blocks** button (obviously, settings that are necessarily gene-specific, such as gene names or intergene definitions, are not copied).

"Genome" box

The options here determine the genome from which **GB→TNT** should retrieve sequences. If the "**ANY**" option is chosen, the accessions belonging to any genome (or even those with no genome specification) will be processed.

"Other" box

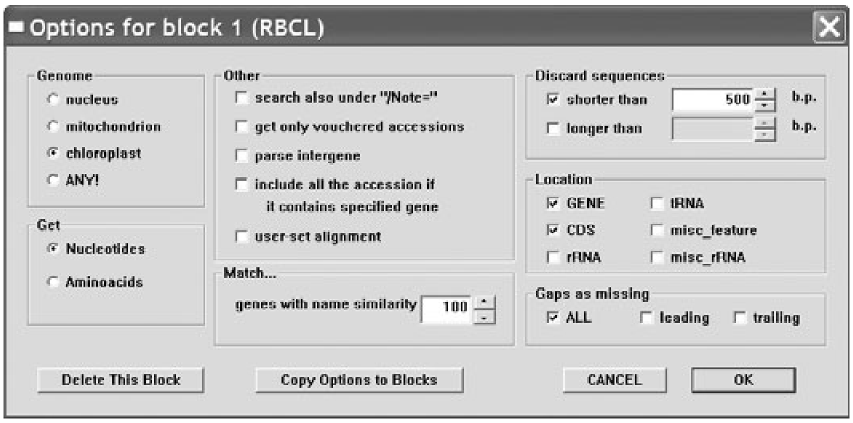
These options determine general settings. By ticking on "**Get only vouchered accessions**", **GB→TNT** keeps only those sequences for which a voucher is specified in the line "/specimen_voucher" of the SOURCE locus. The option to "**Include all the accession if it contains the specified gene**" may be useful to keep not only the chosen gene but also the complete accession that presents the genomic region indicated in the "**Gen/Prod**" box of the "**Edit Project**" dialog. It is also possible to retrieve regions between genes, with "**Parse intergene**"; in this case two different gene or product names (to act as limits) must be defined. The order of the genes should be the same as that in the GB file. The sequences of the flanking genes can either be kept or excluded.

"Discard sequences" box

This box can be used to define higher or lower bounds for the length of sequences to be retrieved. This is useful to remove very incomplete sequences and/or very long sequences (which may indicate a problem in the retrieval of the sequence or a mistake in GenBank annotation).

"Get" box

This option sets whether to retrieve nucleotide or amino acid sequences. If the second option is chosen, **GB→TNT** will keep the sequence included in the line that starts with "/ = translation" in the "CDS" location feature.


Options for block 1 (RBCL)

Genome: nucleus [ ] / mitochondrion [ ] / [✓ chloroplast] / ANY! [ ]

Get: [✓ Nucleotides] / Aminoacids [ ]

Other:
- [ ] search also under "/Note="
- [ ] get only vouchered accessions
- [ ] parse intergene
- [ ] include all the accession if it contains specified gene
- [ ] user-set alignment

Match...: genes with name similarity 100

Discard sequences:
- [✓] shorter than 500 b.p.
- [ ] longer than b.p.

Location: [✓] GENE, [✓] CDS, [ ] rRNA, [ ] tRNA, [ ] misc_feature, [ ] misc_rRNA

Gaps as missing: [✓] ALL, [ ] leading, [ ] trailing

Fig. 3. "Block options" dialog; this is used to set options specific for each gene or block of data.

"Match" box

This option can be used to detect spelling errors in gene or product names. With a name similarity of < 100, a Needleman–Wunsch algorithm is applied to the gene names, so that similar but not identical gene names can be identified. The *similarity* is calculated as $1 - E/L$ (where E is the edit cost of editing the sequence names, and L is the length of the shortest sequence). This option must be used with caution, given that sometimes two different genes may differ only in a single character (e.g. *rps12* vs. *rps15*).

"Location" box

In this box, the user chooses where in the GB file the program will look for the names defined in the "**Gen/Prod**" box of the "**Edit Project**" dialog. The most common locations specified in GB files can be chosen (**GENE**, **CDS**, **tRNA**, **rRNA**, **misc_feature** and **misc_rRNA**). Once **GB→TNT** finds one of these identifiers, it looks for the **Gen/Prod** names in the lines that start with "/gene=" or (when CDS has been requested) "/product=". Optionally, **GB→TNT** can search for the product name in the lines that start with "/note=". Several locations can be chosen together. If **GENE** is chosen, the interval(s) of the sequence to be kept will be those defined in the "gene" headerline of the GB file (e.g.: "gene <1..2521"). If the option **CDS** is chosen instead, the interval(s) of the sequence to be kept will be those defined in the "**CDS**" headerline of the GB file (e.g. "CDS 1..1225,1501..2521"). This last option allows the user to keep as nucleotides only the coding part of the sequence. Alternatively, the user can choose to retrieve the coding sequence as amino acids instead of nucleotides. This is defined in the "**Get**" box of this same dialog.

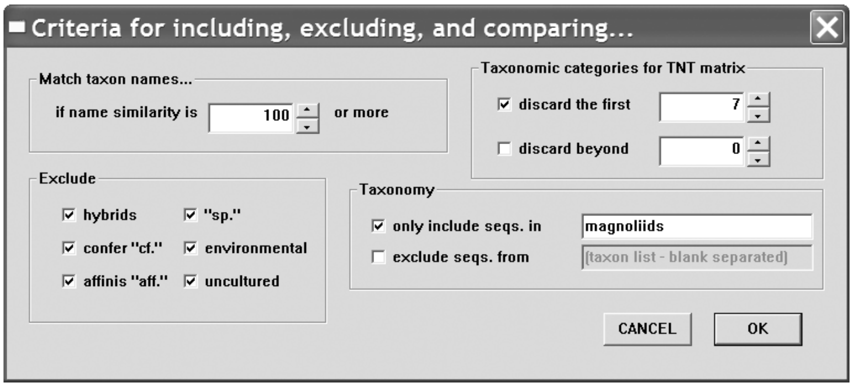
If more than one qualifier is chosen at the same time the program will keep the sequence interval that is found first in the GB file (given that there is a match with the name defined).

Taxonomy settings

A major advantage of using **GB→TNT** to generate **TNT** matrices is the possibility to add all the GB taxonomy to the species names. The new version of **TNT** has a series of tools to make use of this information (see below). Several other options related to the taxonomy are grouped in the **Taxonomy** menu of **GB→TNT**. With these, it is possible to use taxonomic filters for the sequences (e.g. keeping only the sequences belonging to Piperaceae and/or excluding the sequences belonging to Laurales), to define whether species that are provisionally identified in GB (cf., sp., aff.) or environmental samples or uncultured specimens are to be included, and to set the number of taxonomic categories to include in the final **TNT** matrix (Fig. 4).

Given that GB taxonomy is not authoritative and is often outdated, **GB→TNT** provides the option to redefine the taxonomic position of certain species or higher classes. This can be done by selecting **Taxonomy/Change taxonomy**. This is a simple search-and-replace (with the proviso that, in the case of species-level changes where the genus name is modified, the corresponding part of the taxonomy is changed as well). Hence, multiple changes can be performed by just repeating the procedure. Note that this search-and-replace changes the GB files themselves (i.e. all of the GB files included in the directory where the project is located), and does not affect file parsing per se (of course, as the GB files are updated, file precedence—see below—dictates that the GB files must be re-processed). Caution must be taken when input format files other than GB are included in the project because no change will be done to those other format files. Alternatively, the taxonomy can be subsequently changed from **TNT** itself (see below), and saved to a file for future reference (this also allows testing alternative taxonomies).

In the **Taxonomy** menu it is also possible to set whether single or multiple sequences are to be kept for


Criteria for including, excluding, and comparing...

Match taxon names...
if name similarity is 100 or more

Taxonomic categories for TNT matrix
☑ discard the first 7
☐ discard beyond 0

Exclude
☑ hybrids ☑ "sp."
☑ confer "cf." ☑ environmental
☑ affinis "aff." ☑ uncultured

Taxonomy
☑ only include seqs. in magnoliids
☐ exclude seqs. from (taxon list - blank separated)

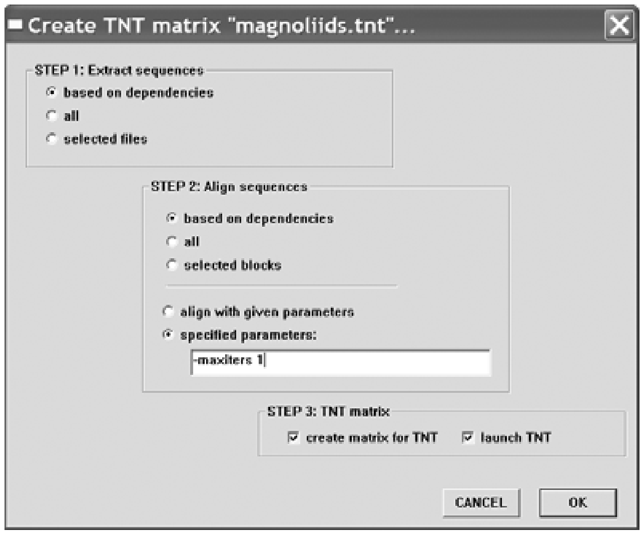
CANCEL OK

Fig. 4. "Inclusion/exclusion" dialog (**Taxonomy** menu).

each block. By default **GB→TNT** keeps only one sequence per species (the longest), which is what most taxonomic studies will need. The "multiple sequences" option keeps the accession number as part of the name. Retaining multiple sequences will usually make no sense when multiple blocks are defined, as in that case it will be impossible to match sequences among different blocks (except in the unlikely case that they belong to the same accession).

Building the TNT matrix

Once all the options are set, the next step is the processing of the GB files (Fig. 5). To do so, **GB→TNT** follows three steps. The first is to actually retrieve the sequences from GB files, for which it considers the settings defined for each block (discussed in the preceding sections). At the end of this step, **GB→TNT** creates a single Fasta file for each block. The second step is the alignment of the Fasta files. The program does not produce the alignments by itself but calls an alignment program. Thus, you need to install **Mafft** and/or **Muscle**, or another alignment program of your preference. It is very common that once the alignments are created the user may want to visually inspect the alignments and, if there are some misaligned sequences, exclude (or reverse-complement) them and proceed to realign. Alternatively, the user may want to trim part of the alignment and keep the modified alignment as a block in the **TNT** matrix. Both actions can be done by calling **BioEdit** (Hall, 1999) from **Project/Check with BioEdit**. Once the user modifies the alignment, **GB→TNT** will ask whether to mark the file for


Create TNT matrix "magnoliids.tnt"...

STEP 1: Extract sequences
- [✓ based on dependencies]
- all
- selected files

STEP 2: Align sequences
- [✓ based on dependencies]
- all
- selected blocks
- align with given parameters
- [✓ specified parameters:] -maxiters 1

STEP 3: TNT matrix
- [✓ create matrix for TNT] [✓ launch TNT]

Fig. 5. "Build matrix" dialog (**Project** menu).

realignment, in which case **GB→TNT** will degap all the sequences of the chosen*.aln* file and copy it onto the corresponding Fasta file. Alternatively, the user may choose to keep the modified alignment as it stands, to be included as a block in the final **TNT** matrix.

The final step is the generation of the **TNT** matrix. Each block in the **TNT** file will correspond to one block of the project. At this point, the program checks for similar names (e.g. *Aristolochia_pistolochia* and *Aristolochia_pastolochia*) using a string-comparison algorithm similar to that used for misspelled gene or product names (note that while the similarity threshold to consider two gene or product names as identical is defined separately for each block, the similarity threshold to consider two taxon names as identical is global and applies to the entire project). Using this option, typing errors can be identified. Finally, **GB→TNT** creates a **TNT** matrix that is ready to run. The **TNT** matrix is always called *project_name.tnt* and it includes (in addition to the data themselves) the settings necessary to process the taxonomy, memory settings, names of the blocks, etc. **TNT** will automatically root the trees based on the taxonomy, in such a way that among the groups resulting from the first split in the taxonomy, the one with fewest terminal taxa is displayed as sister to all the rest of the taxa.

Technical details

Alignment programs

The first time **GB→TNT** is run, it requests from the user the location of the executable files for the alignment programs (this can be subsequently modified from the program with "**Options/Location of...**"). In addition you have to define which alignment program **GB→TNT** will use by default in "**Options/Align with...**". **GB→TNT** will call the alignment program with their default options (i.e. just specification of input/output files). The user can specify different alignment parameters; note that the parameters to be specified do not include (in the case of **Muscle** and **Mafft**) the input/output specification, which is always added by **GB→TNT** to the list of parameters. However, if you are using an alignment program of your choice for the alignment, you must specify the input and output files yourself (as **$infile** and **$outfile**; **GB→TNT** replaces these strings with the corresponding file names). It is also possible to use different alignment programs and/or alignment parameters for each of the different blocks (these options are saved as part of the project, so that they do not need to be redefined when some of the input GB files are to be realigned). Of course, some users may prefer to run the analysis with **POY** (Varón et al., 2010), which requires no prior alignment. Even in such cases, generating the data sets with **GB→TNT** will facilitate later export of **POY** results into **TNT**, for taxonomic diagnosis and evaluation as described below.

Taxon naming conventions

Given that the *GB accession numbers* are (by default) retained for each sequence along the pipeline, it is possible to keep track of the source for each of the sequences included in the final **TNT** matrix, thus making studies from **GB→TNT** fully self-documented and easily reproducible.

The *name* of the species is identical in each block (in case **GB→TNT** established some equivalencies, based on highly similar names, the "valid" name, that is the one-first encountered, is used throughout). The accession number is (normally) different in each block; for establishing taxon identity, **TNT** uses only the name of the species, not the accession number or the taxonomic information. This is because **GB→TNT** adds a quadruple underscore to the name of the species, and **TNT** uses only the string preceding that quadruple underscore for name-matching. The *taxonomy* is included in the taxon names, after an "@" symbol; like the accession number, the taxonomy is located beyond the quadruple underscore, and is thus not used by **TNT** for the purpose of name-matching between different data blocks. A caveat here: this approach will not detect homonyms from different Kingdoms, which must be identified separately.

Note that **TNT** stores in memory the name of the first occurrence of each taxon, so that if the first block contains:

Aristolochia_pistolochia____DQ53206_@Eukaryota

a second block contains:

Aristolochia_pistolochia____DQ182332_@Eukaryota

and a third (e.g. morphological) block contains the shorter:

Aristolochia_pistolochia

the three entries will be considered to correspond to the same terminal, and **TNT** will store the name for that terminal as the first string found (i.e. accession DQ53206). Therefore, if a species in a morphological block corresponds to species already defined in previous (molecular) blocks, it is not necessary to explicitly add the taxonomy to that species; the taxonomy is required only for the first entry (by the same token, you must ensure that the first entry for the species in the **TNT** matrix does have the taxonomy; inverting the order of the blocks in the *Aristolochia* example just given would store the name without the taxonomy!).

Last, using an ellipsis to end a taxon name in a **TNT** block is interpreted as assigning immediately following character states to all the previously described species which match the string provided; thus, listing the taxon in the last (morphological) block as

Aristolochia_pistolochia…

would be identical to the above example, but using instead

Aristolochia_…

would cause subsequently specified character states to be assigned to each of the species of the genus *Aristolochia* included in previous blocks. This can be used to add morphological data for groups of species with economy of keystrokes.

File precedence

GB→TNT keeps track of the modification dates and times of all the files it uses in the process; when the options for some blocks are modified and the matrix is built again, the program will retrieve or align again only the necessary files. This behaviour can be overridden, as the user can also specify one by one the files that are to be retrieved/processed. If in doubt, choose "**all**" (under the menu option **Project/Build Matrix**), for both retrieval and alignment. Care must be taken when some alignment(s) has(ve) been modified with **BioEdit**. In that case, to preserve the changes, you either (i) choose the "**based on precedence**" option in the first and second step in the **Project/Build matrix** dialog, or (ii) choose "**selected files**" and "**selected blocks**" and select the blocks with manually modified files to be skipped. Note that because most of the **Taxonomy** settings are global, and do not affect the data blocks individually, changing them usually does not require that GB source files are reprocessed and realigned; this simply requires re-merging the files already processed for creation of a new **TNT** matrix.

Memory usage

Because **GB→TNT** is intended to directly launch **TNT**, it is designed to have low memory requirements instead of high speed, thus leaving more resources available to **TNT**. The most significant memory saving comes in that **GB→TNT** does not store in memory the sequences it retrieves; to merge the (unaligned) files and retain the longest sequence, **GB→TNT** does two passes for each file. In the first pass, it records only the species name and the length of the sequence (instead of the sequence itself); in the second pass, **GB→TNT** reopens the file and copies the longest occurrence of each species from one temporary file to the other. This means that (i) a number of temporary files are created as part of the process, and (ii) the processing is not as fast as it could be. As the retrieval of sequences and the matching of taxon names are not very time-consuming operations (compared with the times required for alignment or phylogenetic inference), it is preferable to privilege memory usage.

A worked example

This sections shows how to create, with **GB→TNT**, a TNT matrix including *rbc*L and *mat*K for all the species of magnoliids available in GenBank, with at least 500 bp sequenced. The GB files, *magnoliids_rbcl.gb* and *magnoliids_matk.gb*, are included together with the **GB→TNT** package; these two files were generated and downloaded directly from the NCBI homepage.

We will start first by generating a new project (**File/New Project**, which opens the "**Edit project**" dialog). We will rename the first block as *rbc*L, and indicate to the program where the GB file to be parsed is [by clicking on the "**find file(s)**" button]. After that we will define the name of the gene we want to retrieve (*rbc*L in this case) in the "**Gen/Prod**" box. The options for this block are set with the "**Options**" button (below the "**Gen/Prod**" box); because *rbc*L is a chloroplastidial gene we will click the "**Chloroplast**" option in the "**Genome**" box; the default option of "**Get**" is "**Nucleotide**", so no modification is needed there.

To generate the second block we press the "**Add block**" button of the "**Edit Project**" window. As in the case of the *rbc*L block we will rename the block, define where the GB file to be parsed is, and write the name of the gene in the "**Gen/Prod**" box. Here, we will include two ways of naming the same gene, *mat*k and maturase k. As the second is a product name, we must also choose the **CDS** option of the "**Location**" box (this is there where the product names are defined in GB files).

After that, the "**Edit project**" dialog can be closed by pressing "**OK**" and the project can be saved (with **File/Save**). We then define the taxonomy options (**Taxonomy/Inclusion-Exclusion Criteria**). Here we will indicate to the program that we want to exclude the first seven categories from the taxonomy that will be added to each terminal in the **TNT** file (because those seven categories are common to all the sequences; alternatively, redundant categories can be wiped out by TNT itself) and that we just want species belonging to magnoliids (in case species for other groups were infiltrated in the GB file).

The alignment program to use can be defined from the menu "**Options/Align with...**" (or individually for each block); in this case, we will select **Muscle** as the aligner for all the blocks (from **Options/Choose Default Aligner**). Finally, we will proceed to build the matrix (**Project/Build Matrix**). For alignment ("**STEP 2**" box), we will set "-maxiters 1" (a faster **Muscle** option, as we do not need high accuracy for this example). To generate the **TNT** matrix we will leave the "**STEP 3**" options as default (i.e. "**Create matrix for TNT**" and "**Launch TNT**" selected). After pressing "**OK**", **GB→TNT** parses the GB files, calls **Muscle** to perform the alignment, and generates the **TNT** matrix, without the need for human intervention. With the settings shown, the matrix consists of 1376 taxa and 4247 characters. All the work needed to assemble this large data set, from the downloading of the GB files to the last step in the creation of the matrix (including alignment), could be completed in a couple of hours.

Evaluation of results with new TNT options

The matrix resulting from the previous example can be analysed with TNT; for the present example, we simply used the default options of the *xmult* command (roughly equivalent to the default options of **Analyze/New Technology Search**). To minimize the number of misplaced taxa, we deactivated (from **Data/Character Groups & Blocks/Active Blocks**) all those taxa scored for only one of the two genes; thus, 761 taxa remain active (we pruned, after the search, an additional five misplaced taxa, to show results more clearly). Figure 6 shows several possible ways to display the results. The tree of Fig. 6a is the complete tree, with the Orders coloured.

One of the problems in the analysis of large data sets is that it becomes difficult to visualize results. In the analysis of the large eukaryote data set of Goloboff et al. (2009), this problem was solved by means of **TNT** scripts which assess the fit, and scripts which colour branches, for specified taxonomic groups. This allows for some automation, but the scripts were both slow (as they have to be interpreted by **TNT**) and hard to use. The new version of **TNT** incorporates several native options to handle the taxonomy embedded in the taxon names; among other possibilities, the program can now evaluate the fit of the results to the taxonomy (**Trees/Taxonomy/Report Fit of Groups**), either for selected taxonomic groups or for categories. The categories can be recognized by the ending of the category names (e.g. ales, oideae, aceae). The fit is calculated as in Goloboff et al. (2009, p. 215); for a reference taxonomic group, the closest group in a tree is that for which the number of extraneous taxa that have been included in the group, and the number of taxa of the group that have been excluded, sum to a minimum (if above a certain threshold, which can be changed by the user, the taxonomic group is concluded as absent). TNT allows approximate group recovery because, when only one or a few species of a large group are misplaced, it is much more useful—in practical terms—to indicate that the group is "almost" recovered than it is to simply indicate "non-recovery".

The fit can be printed on the tree branches (**Trees/Taxonomy/Label Tree Branches**) or given as a list (**Trees/Taxonomy/Report Fit Of Groups**, Table 1). In the latter case, if there is a perfect match **TNT** indicates *Mono*; if there is a non-perfect match **TNT** indicates the number of taxa each taxonomic group

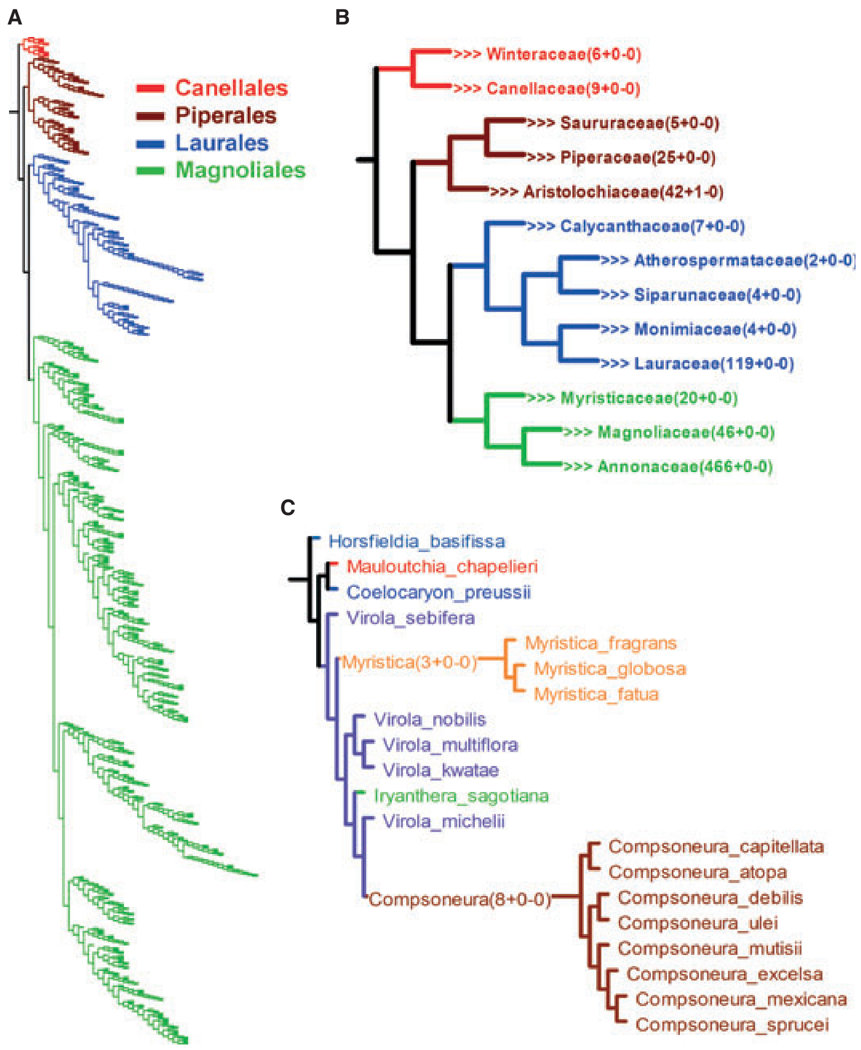


Fig. 6. Trees for the Magnoliales, coloured by **TNT** (based on the GenBank taxonomy, incorporated into the taxon names by **GB→TNT**). Tree diagrams such as these can be obtained with a few mouse clicks, and require no further editing. (a) The complete tree (751 taxa), with the four orders shown in different colours. (b) All the families labelled and contracted (the arrows indicate that those groups can be expanded on display; double left-clicking on a node contracts it or expands it; a double left-click on the root contracts or expands all nodes simultaneously). (c) One of the families, Myristicaceae, showing each genus in a different colour, and branches corresponding to a genus labelled as such.

contains, the number of extraneous taxa added to the group (preceded by a plus sign), and the number of taxa from the group which have been placed outside of the group (preceded by a minus sign).

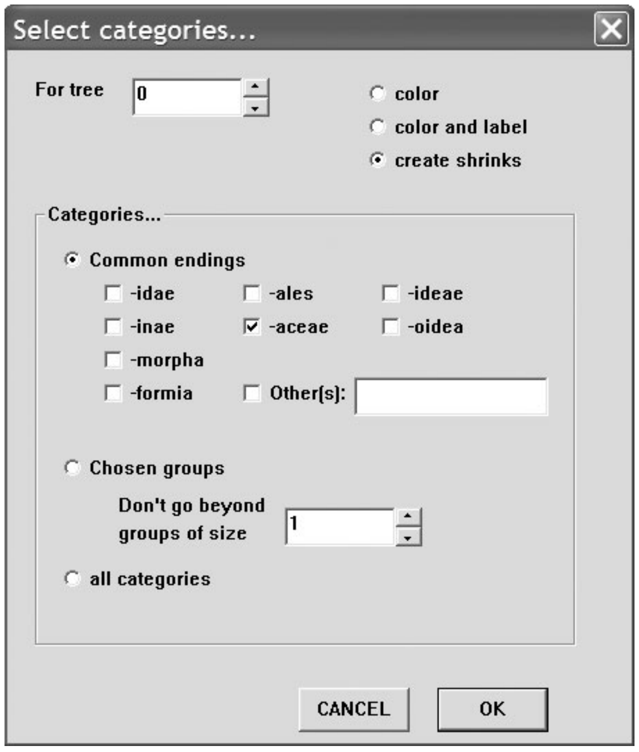
The above function allows easy identification and manipulation of those terminal taxa which, in the results, do not belong to the expected taxonomic groups. For any specified set of taxonomic groups, such terminal taxa can be placed in taxon groups with **Trees/Taxonomy/Report Fit Of Groups** (or with the "&" and "|" options of the *taxonomy* command). Once these problematic terminals have been identified, they can be removed from trees or consensi, or further inspected for possible misidentifications or contaminated sequences.

The taxonomy can also be easily visualized on the results as tree-diagrams (with the menu option **Trees/Taxonomy/Show Groups On Tree**, see Fig. 7). One possibility is to shrink those groups that (approximately) fit the taxonomy. The groups that are not present in the tree are not shrunk and remain as terminals in the tree. Until the shrinks are explicitly deactivated (e.g. with the command "*tshrink* **-**.;", or by a double left-click on the root node with the tree unlocked), any tree plotted which contains those groups defined for shrinks will be displayed in their shrunken form. The shrinkings are very useful to evaluate the relationships between higher clades on the tree. For instance, it is very easy to generate a "tree of families" in which each family present in the tree is labelled

Table 1
Output of **TNT**, reporting fit for the orders and families for the example magnoliid data set (**Trees/Taxonomy/Report Fit Of Groups** option)

```
Laurales (137 taxa): (MONO)
 Lauraceae (119 taxa): (MONO)
 Atherospermataceae (2 taxa): (MONO)
 Monimiaceae (4 taxa): (MONO)
 Hernandiaceae (1 taxa): (irrelevant)
 Calycanthaceae (7 taxa): (MONO)
 Siparunaceae (4 taxa): (MONO)
Magnoliales (535 taxa): (MONO)
 Annonaceae (466 taxa): (MONO)
 Magnoliaceae (46 taxa): (MONO)
 Degeneriaceae, Degeneria (1 taxa): (irrelevant)
 Myristicaceae (20 taxa): (MONO)
Canellales (15 taxa): (MONO)
 Winteraceae (6 taxa): (MONO)
 Canellaceae (9 taxa): (MONO)
Piperales (74 taxa): (MONO)
 Aristolochiaceae (42 taxa): (+1–0,0.9762)
 Piperaceae (26 taxa): (+0–1,0.9615)
 Saururaceae (5 taxa): (MONO)
```

All the groups are monophyletic, except Aristolochiaceae and Piperaceae (for which the number of extraneous and excluded taxa in the closest group found is reported). The number of taxa belonging to each group is indicated by **TNT** in parentheses, after the group name.


Select categories...

For tree: 0

color
color and label
[✓ create shrinks]

Categories...

[✓ Common endings]
-idae; -ales; -ideae; -inae; [✓ -aceae]; -oidea; -morpha; -formia; Other[s]:

Chosen groups
Don't go beyond groups of size: 1

all categories

Fig. 7. The TNT dialog for colouring branches as in Fig. 6 (available from **Trees/Taxonomy/Show Groups On Tree**).

(Fig. 6b). It is also easy to additionally colour the taxonomic groups, either those belonging to a given category or those selected individually.

For instance, in Fig. 6a,b each Order was printed with a different colour. To both shrink and colour groups, you need to effect two actions, first defining the nodes to shrink as families (ticking on "**create shrinks**" and "**-aceae**", then "**OK**"), then colouring and labelling orders (ticking on "**color and label**" and "**-ales**", then "**OK**"). Once the second (coloured) tree-diagram is shown, it can be saved as a metafile by pressing "**M**". To select second- or third-order splits of a given taxonomic group (as in Fig. 6c), clicking on "**chosen groups**" displays the (taxonomic) tree, and repeatedly left-clicking on a node selects the splits of the subsequent level (with ctrl + left-click, the intermediate categories are included in the list; right clicking brings the selection down). Selecting N-order splits avoids the need to specify the splits one by one, making it easy to select, for example, all the genera of a family, or all the families of an order.

In the case of extremely large trees (e.g. as in Goloboff et al., 2009), it is possible (in Windows version) to not print the names of terminal taxa and resize the trees so that they fit the screen (by pressing "**N**" and "**F**", respectively; these are toggles). If the groups are properly coloured, then this produces diagrams which are informative at a glance.

All these actions can also be performed by commands. The complete set of commands needed to colour a tree held in RAM as shown in Fig. 6 is:

```
ttag=;
taxonomy <0 =aceae_;
taxonomy :0 =ales_;
taxonomy *0 =ales_;
ttag & example_one.svg colors;

tshrink -.;
pruntax 0/+.-:Myristicaceae;
ttag-;
ttag=;
taxonomy :0 \2 Myristicaceae;
taxonomy *0/3 Myristicaceae;
ttag & example_two.svg colors;
```

these commands produce files *example_one.svg* and *example_two.svg*, two SVG files (Scalable Vector Graphics), which can be opened with most web browsers. First, the *ttag* command instructs **TNT** to concatenate the branch labels (see Goloboff et al., 2008: 783; and documentation for **TNT**) of subsequently plotted trees. For the specified tree (tree 0, in the example), the "<" option of the *taxonomy* command shrinks (as discussed above) those groups which correspond to the taxonomic groups specified in the subsequent list; the "=" symbol will place in the list of taxonomic groups to shrink all those groups which end with "aceae" (i.e. families). The ":" option of *taxonomy* displays, in different colours, each of the subsequently specified taxonomic groups (because the *ttag* option is activated, the branch labels corresponding to the colour codes will be stored for subsequent use, instead of displayed); as in the previous option, "=*ales_*" will colour those taxonomic groups ending with "ales" (i.e. orders). Then, the "*" option labels those groups of the taxonomy effectively found in the trees specified. Because the *ttag* option continues in effect, instead of displaying the tree, the branch labels are stored and concatenated with previous ones. Last, the *ttag* command is used to save a tree-diagram with the tags stored interpreted as colours. Thus, only five lines of command are needed to produce a diagram exactly like that shown in Fig. 6b. We have avoided merging several of these options into more comprehensive ones; although this admittedly makes the syntax slightly more complex, it gives the user the freedom to apply only some of these steps (e.g. making it easy to produce tree-diagrams either with or without shrunken groups, text labels, or colours).

The last example (Fig. 6c) shows the genera of Myristicaceae only, using different colours for the genera and labelling those genera recovered in the tree (note that there is no branch labelled *Virola*, as that genus does not correspond to any group in the tree; its species, however, are all equally coloured). The first four lines for the second example first deactivate shrinkages and prune from the tree all the taxa not belonging to the Myristicaceae, then clearing and resetting the tags for concatenation. The rest is similar to the previous example, except that specifying the groups as "*\2 Myristicaceae*" indicates that the splits up to two levels below the Myristicaceae are to be coloured (excluding intermediate categories, in this case, those one level below), and "*/3 Myristicaceae*" labels all taxonomic categories (including intermediate ones) up to three levels below Myristicaceae.

In cases where it is desirable to test alternative taxonomies, instead of changing the taxonomy in the GB files with **GB→TNT**, an existing taxonomy can be changed with **TNT** using the *taxname* command. Additionally, Windows versions of **TNT** also allow editing the taxonomy graphically (first saving the taxonomy tags, by allowing multiple tags and plotting the taxonomy as a tree, then editing tags with **Trees/Multiple Tags/Edit Tags**, and finally transferring the tags to the taxonomy, with the *copytree* command or with **Settings/Taxonomy Settings/Copy Tree-Tags to Taxonomy**). Once the new taxonomy has been defined, it can be saved as a tagged-tree and subsequently read back.

Given that the **GB→TNT** pipeline is mostly automatic, there is an increased risk that some sequences are in the wrong sense. To help prevent this, **TNT** now allows detection of reverse-complements, by making pairwise comparisons between the sequences (one block at a time), with the *rcompl* command. For each pair of sequences to be evaluated, the Needleman–Wunsch edit cost E of the sequences as they are, and the edit cost R with one of the sequences reverse-complemented, are compared. If $E < R$, and $(R - E)/E < 0.2$, then the two sequences are concluded to point in the same direction; if $E > R$, and $(E - R)/R < 0.2$, the sequences are concluded to point in opposite directions (the criterion is conservative, in that differences falling between the $+0.2$ and -0.2 interval are considered inconclusive). The program uses a judicious choice of pairwise comparisons (e.g. avoiding comparison between two sequences which have already been found to point in the same direction as another, and skipping calculation of the edit cost when the Manhattan distance between the aligned sequences is low), so that this criterion can be applied to relatively large data sets.

Acknowledgements

We thank the Willi Hennig Society for subsidizing the program TNT and making it freely available, and Salvador Arias, Gonzalo Giribet, Damon Little, Marcos Mirande, and Claudia Szumik for comments on the manuscript and/or the program. The research for this paper was carried out at the Instituto Superior de Entomología "Dr. Abraham Willink" and the Instituto Argentino de Estudios Filogenéticos. Financial support from FONCyT 1314 (to P.A.G.) and PIP CONICET 0696 (to Mercedes L. de Grosso) is greatly appreciated.

References

Chevenet, F., Croce, O., Hebrard, M., Christen, R., Berry, V., 2010. ScripTree: scripting phylogenetic graphics. Bioinformatics 26, 1125–1126.

Edgar, R.C., 2004. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5, 1–19.

Goloboff, P.A., Farris, J.S., Nixon, K. 2003. Tree analysis using new technology Version 1.1. Available from http://www.zmuc.dk/public/phylogeny/TNT.

Goloboff, P., Farris, J., Nixon, K., 2008. TNT: a free program for phylogenetic analysis. Cladistics 24, 774–786.

Goloboff, P.A., Catalano, S.A., Mirande, J.M., Szumik, C.A., Arias, J.S., Källersjö, M., Farris, J.S., 2009. Phylogenetic analysis of 73 060 taxa corroborates major eukaryotic groups. Cladistics 25, 211–230.

Hall, T.A., 1999. BioEdit: a user-friendly biological sequence alignment editor and analysis program for Windows 95/98/NT. Nucleic Acids Symp. Ser. 41, 95–98.

Huson, D., Richter, D.C., Rausch, C., Rupp, R. 2011. Dendroscope vers. 3.0.13 beta. Available from http://ab.inf.uni-tuebingen.de/software/dendroscope/.

Katoh, T., 2008. Recent developments in the MAFFT multiple sequence alignment program. Brief. Bioinform. 9, 286–298.

Page, R.D.M., 1996. TreeView: an application to display phylogenetic trees on personal computers. Comput. Appl. Biosci. 12, 357–358.

Rambaut, A. 2009. FigTree, vers. 1.3.1. Available from http://tree.bio.ed.ac.uk/software/figtree.

Sanderson, M. 2006. Paloverde, a three-D large tree visualization tool. Available from http://loco.biosci.arizona.edu/paloverde/paloverde.html.

Smith, S.A., Beaulieu, J., Stamatakis, A., Donoghue, M.J., 2011. Understanding angiosperm diversification using large and small phylogenies. Am. J. Bot. 98, 404–414.

Stöver, B., Müller, K., 2010. TreeGraph 2: combining and visualizing evidence from different phylogenetic analyses. BMC Bioinformatics, 11, 7.

Vaidya, G., Lohman, D., Meier, R., 2010. SequenceMatrix: concatenation software for the fast assembly of multi-gene datasets with character set and codon information. Cladistics 27, 171–180.

Varón, A., Vinh, L., Wheeler, W., 2010. POY version 4: phylogenetic analysis using dynamic homologies. Cladistics 26, 72–85.