

September 2017

**A guide for the analysis of continuous and
landmark characters in TNT (Tree Analysis
using New Technologies)**

Santiago A. Catalano

Unidad Ejecutora Lillo (UEL FML-CONICET)

Facultad de Ciencias Naturales e IML. UNT

San Miguel de Tucumán - Argentina

sacatalano@gmail.com

Pablo A. Goloboff

Unidad Ejecutora Lillo (UEL FML-CONICET)

San Miguel de Tucumán - Argentina

pablogolo@yahoo.com.ar

CONTENTS

1. INTRODUCTION	4
12. GENERALITIES	4
2.1. Interacting with the Program.	4
2.3. Command Truncation	4
2.4. Data “In Memory”	4
2.3. Numeration.	5
3. CONTINUOUS CHARACTERS	5
3.1. Introduction	5
3.2. Input Data.	6
3.2.1. Standardization	7
3.2.2. Fitting Ranges	7
3.3. Optimization.	7
3.4. Dependence, Scale, Ratios, Etc.	7
3.5. Support.	8
4. LANDMARK DATA	8
4.1. Concept of Character in Landmark Data.	8
4.2. Order of Landmark Configurations	9
4.3. Landmark Settings and Info	9
4.4. Handling Data	10
4.4.1. Data Input.	10
4.4.2. Outputting Results.	10
4.4.2.1. Modified Matrices	
4.4.2.2. Ancestral configurations	
4.4.2.3. Exporting Configurations in <i>tps</i> Format	
4.4.2.4. Drawings	
4.4.2.5. Export Tree with Branch Lengths	
4.4.3. Missing Data	
4.5. Algorithms to Optimize Landmark Data.	13
4.5.1. Optimization of Observed States.	13
4.5.2. Optimization Using Grids.	14
4.5.3. Optimization Using Nested Grids.	14
4.5.4. Combining Approaches.	14
4.5.5. Grid Shaking, Grid Shrinking, Cell Skipping.	15
4.5.6. Node by Node Iterative Refinement.	15

4.6.- Weighting/Scaling/Factors/Standardization	16
4.6.1. Weights	16
4.6.2. Factors	17
4.7. Mapping Shapes. Lines. Numbers. Ancestral/Descendant Shapes	
Superimposed. Connectors, Etc.	18
4.7.1. Visualization of Landmark Optimization	18
4.7.2. 3D Visualization	21
4.8. Ambiguity in Ancestral Landmark Reconstruction	22
4.9. Score Calculation	22
4.10. Superimposing Configurations	22
4.10.1. Pairwise Superimposition	23
4.10.2. Two-Point Registration	23
4.10.3. Using a Tree as Guide	24
4.11.- Phylogenetic Searches	25
4.11.1. Generalities	25
4.11.2. Searches Combining Multiple Configurations	26
4.11.3. Group Support	26
4.11.4. Searches and Dynamic Superimposition	27
4.11.5. Implied Weighting	28
4.11.6. Fine Tuning of Searching Algorithms for Landmark Data	30
4.12. Calculation of Phylogenetic Signal	30
4.13. Macro (Scripting) Expression for Landmark Data	30
5. BIBLIOGRAPHY	32
6. LINKS	32

1. INTRODUCTION

This guide is intended to help TNT users to analyze continuous characters and landmark data. It assumes the reader already has some basic knowledge of how to use the program. The first TNT version that allows handling continuous characters was released in 2004, while the first version that allows analyzing landmark data was released in 2009. Versions subsequent to the early releases have included many new commands and functions to analyze continuous and landmark characters. However, until recently, many of the functions and commands in TNT could not be executed on landmark data. In December 2015 the landmark characters were fully and seamlessly integrated with all of the functionality in TNT. In particular, it became possible to perform phylogenetic searches in datasets containing landmark data with the same native TNT commands used to analyze other types of data (searches for landmark data prior to December 2015 required the use of special scripts). The theoretical basis and algorithms to analyze continuous characters are described in Farris (1970) and Goloboff et al. (2006). The theoretical basis and algorithms for landmark analysis in a parsimony context are described in Catalano et al. (2010), Goloboff & Catalano (2011), Catalano & Goloboff (2012) and Goloboff & Catalano (2016).

2. GENERALITIES

2.1. Interacting with the Program:

There are two different ways to interact with the program in the Windows version of TNT. One of them is using the GUI ("graphical user interface"), i.e. the drop-down menus. The other is typing the commands in the command line at the bottom of the TNT screen. The Linux and Mac versions do not have a GUI, and then interaction with TNT for Linux and Mac is only through commands. Throughout this guide, both the use of menus and commands will be discussed. Throughout this document, regular TNT commands are shown in *italics*, menu paths in **bold** and macro commands as underlined.

2.2. Command Truncation

TNT allows truncation when typing commands. For instance, typing *lmr* will execute the command *lmreal*. If there is ambiguity in the truncation, TNT will generally solve it by choosing the first command that matches the string provided (in alphabetical order). For instance if you type *lm* TNT will run the command *lmark* and not *lmreal* because the letter "a" is before "r" in the alphabet. The few exceptions to this alphabetical rule are in some commands used very commonly:

e.g. "p" is recognized as "procedure", instead of "pause", the first matching command under alphabetical order.

2.3. Data "In memory"

When the data are read into TNT, the information is kept in RAM memory. Only one "matrix" can be read at a time (but this matrix can have multiple partitions, to be analyzed independently). Some commands modify the data information kept in memory (e.g. realignment of landmark configurations) in such a way that the following commands are executed on the modified matrix. The user can save the modified matrix onto a file using the command *xread** (**Data/SaveData** menu option). Once the data in memory are modified, the only way to restore the original matrix is by re-reading it.

2.4. Numbering

The numbering of characters, taxa, trees, states, etc. always starts from zero, with the last of N elements numbered as $N-1$. The only exception is the numbering of data blocks, which starts from 1 (with 0 representing "the entire matrix", i.e. all the blocks). When entering the commands in the command line, it is possible to enter ranges by placing a period (".") between the extreme values of the range. If the period is preceded by a blank or any non-digit, then the range is taken to start at the first possible element (usually, 0); if the period is followed by a blank or any non-digit, then the range is taken to end at the last possible element ($N-1$). Thus, the symbol "." by itself is equivalent to a list of all the elements.

Example

Deactivate characters 4 to 9.

```
ccode ] 4.9 ;
```

Deactivate from character 3 to the last character in the matrix

```
ccode ] 3. ;
```

Deactivate all characters.

```
ccode ] . ;
```

3. CONTINUOUS CHARACTERS

3.1. Introduction

Continuous characters are always analyzed in TNT as additive characters, using Farris optimization (Farris 1970), with the implementation described in Goloboff et al. (2006). In this context, transformation costs are defined as the numerical difference between states. No discretization step (e.g. gap coding methods) is required. Most of the commands available in TNT for discrete characters are also available for continuous characters. Exceptions are visualization of individual ancestral reconstructions (see below), and counting the number of specific transformations. Continuous characters can be edited with the *xread=* command, and with the menu option **Data/Edit/Taxon**, but not with **Data/Edit/Character**. Continuous characters can be named with the *cnames* command, but their states cannot (i.e. the states can only be numerical values).

3.2. Input data

TNT accept continuous characters in a range from 0.000 to 65.000. Continuous characters can be combined with discrete or landmark characters, but they should always be included first in the matrix, before any other character. TNT accepts three decimals (with “.” as separator for decimals) as values for continuous characters. Space(s) or tab(s) separate(s) the characters.

Example

```
xread
3 5
& [cont]
Sp1 0.02 45.561 3.0
Sp2 0.23 20.123 3.0
Sp3 0.12 43.31 3.0
Sp4 0.12 35.531 2.0
Sp5 0.42 15.361 2.0
;
```

Continuous characters can be entered as single values (the mean value for instance) or as a range. There are two ways to enter a range, with the two extreme values separated by a hyphen or as the central value plus/minus the difference to the limits of the range. This can be used for instance to express the range as the mean plus/minus a standard deviation.

Example

For instance, if the mean is 5 and the Standard Deviation is 0.5, the range can be set as:

5+/-0.5

Internally, TNT considers this as the range 4.5-5.5.

```
xread
& [cont]
2 5
Sp1 12.3      3.0±0.01
Sp2 12.1-12.5 3.0±0.02
Sp3 12.1-12.5 2.0±0.02
Sp4 12.1-13.5 2.125
Sp5 12.1-13.5 2.121
;
proc/;
```

3.2.3. Standardization

To infer phylogenetic relationships, continuous characters should preferably be standardized. Otherwise, characters that differ in the scale of measurement may disproportionately contribute to resolving the final phylogenetic hypothesis. TNT allows standardizing continuous characters so that the maximum range (i.e. the maximum possible difference between any two taxa) of a continuous character equals N steps of a discrete character. The command is *nstates stand N L*, where N is the value to give to the largest original value (and the lowest value automatically set to 0), and L is a list of (continuous) characters to be rescaled. If no character is indicated, the standardization is applied to all continuous characters in the matrix.

3.2.3 Fitting the Range

If a continuous character in the matrix falls outside rescaled the 0.000-65.000, the values are modified to fit in it. If the intention is to perform phylogenetic searches, the characters whose range has been modified, should be standardized.

3.3. Optimization

The command to map continuous characters on a given tree is the same as for traditional characters (**Optimize/Characters/Character Mapping**). This will give the range of optimal states for each ancestral node. When mapping standardized continuous characters, TNT will show the standardized values.

However, it is possible to display the values in the original scale (the one in the datafile) using the command *nstates* [.

In most cases, the optimization of continuous characters determines as optimal states a range of values for several nodes. Hence, the possible number of individual reconstructions (i.e. particular combinations of ancestral values for all the nodes of the tree that produces the optimal score) can be extremely high, even with the precision limit of 0.001 imposed by the implementation in TNT. That is why, unlike the case for discrete characters, it is not possible to visualize each optimal reconstruction for continuous characters using commands or menus. However, this can only be done using *iterrecs* , a macro TNT macro expression that allows “retrieving” these individual reconstructions (if not all, at least a good sample of them). See macro documentation for a complete description of this expression ([iterrecs](#)).

3.4. Dependence, Scale, Ratios, IW, Etc.

TNT has no special treatment for the possible existence of dependence among continuous characters. The only advice that can be offered is to use common sense, the same common sense that should be used in the analysis of discrete characters. For instance, if the user wants to consider the variation in size of ten different structures as different characters, and the whole body of the species has changed, you will be strongly overweighting the evidence provided by the change in size. An discussion about the use of ratios in phylogenetics in Mongiardino et al. (2010). Keep in mind that this dependence only affects tree-choice (i.e. searches); this problem is not equally relevant when the continuous characters are only to be mapped to be mapped on a given tree.

3.5. Support

Optimal trees obtained from the analysis of continuous characters tend to be more resolved than trees obtained from discrete characters. This is because two taxa are rarely identical for continuous characters, and some nodes can be supported by only fractions of steps. Strict consensus are also usually more resolved, given that it is less likely to have exact ties among characters. As a consequence, some groups present in the strict consensus may have a very low support. That is why it is imperative to calculate support measures for matrices

with continuous characters, eliminating groups with very low supports. As indicated below, the same holds for landmark data.

4. LANDMARK DATA

4.1. Concept of Character in Landmark Data

Defining what "a character" represents is not an easy task in phylogenetics. On a superficial analysis, character definition in sequence data might seem easy: each site (base) in a sequence represents a character. However, sequences suffer inversion, deletions, insertions and duplications that complicate the definition of a character. In morphological characters the definition of a character can be obvious in some cases (presence/absence of spinal column), but in general the definition of a character must be subject to additional considerations. In the end, there are no radical differences with the type of decisions that must be made for the phylogenetic analysis of discrete characters. Keeping that in mind, TNT makes the practical decision of considering each configuration as a different character by default, making the contribution from each configuration roughly equivalent to that of a discrete character. This is intended more as a practical approach to the problem than as a conceptual definition. This equivalence is obtained by standardizing the configurations so that the contribution of each configuration is similar to a discrete character irrespective of the scale of the configurations or the number of landmarks (see section on Standardization). However, considering different weights and factors, the user may change the equalization to discrete characters, either up- or down-weighting landmark configurations. Using different weighting schemes and standardizations, the user may choose to consider every individual landmark as a different character, or (going to the other extreme) a set of several different configurations as an individual character. TNT is thus agnostic in this regard, providing the user with a range of options, from treating individual landmarks as several characters, to treating several configurations as a single character.

4.2. Order of Landmark Configurations

When landmark configurations are combined with other kind of characters, the numbering is consecutive, considering every configuration as a different character.

Example. Map the first configuration on the second tree in memory in a dataset where the first ten characters are continuous characters and the following three characters are landmark configurations.

lmark map 1/10 ; (Remember that the first character and tree is always 0)

Example. Deactivate the 12th character, i.e. the second landmark configuration, the command would be

ccode] 11 ;

4.3. Landmark Settings and Info

In Windows menu-based versions of TNT, commands and settings for landmark data can be defined from two Windows dialogs . General and search settings are defined in **Settings/Landmarks**. The menu option to define settings and commands for superimposition of landmark configurations is **Data/EditData/Landmark_Alignment**.

-Showing landmark settings

lmark opt: show the current landmark settings

lmark factors: show the current factors.

lmark dims: show the number of dimension of each configuration. If the matrix combines different sources of evidence: 0= discrete (adimensional), 1= continuous (lineal), 2= 2D landmarks, 3= 3D landmarks.

lmark numlands: give the number of landmarks for every configuration

4.4. Handling Data

4.4.1. Data Input

The easiest way to include several landmark configurations (i.e. several shapes) in a TNT file is to use separate blocks for every configuration.

Example.

```
xread
2 4
&[landmark 2d]
Sp0 0.552,0.552 0.552,0.552 0.552,0.552
Sp1 0.552,0.552 0.552,0.552 0.552,0.552
Sp2 0.223,0.552 0.552,0.552 0.552,0.552
Sp3 0.231,0.552 0.552,0.552 0.552,0.552
&[landmark 3d]
Sp0 -0.031,234.423,10.003 1.003,0.3,0.453 -12.03,0.147,56.8
Sp1 -0.031,234.233,12.003 2.003,0.3,0.713 -12.89,0.258,56.678
Sp2 0.123,244.453,14.003 4.003,0.3,0.713 12.77,0.003,56.673
Sp3 0.343,254.111,12.003 4.003,0.3,0.103 12.45,0.003,56.67
;
proc/;
```

Alternatively, you can separate the configuration by pipe symbols (" | "), instead of placing each in a separate block . Notice that only the number of configurations and taxa should be specified in *xread*, NOT the number of landmarks (which is calculated automatically).

TNT can also import landmark data from files in tps format. It is possible to automatically merge multiple tps files, where each file includes all the species from a landmark configuration. The species name is extracted from the ID lines. The menu option to read tps files is **File/MergeImportData/ImportTpsFiles**. Note that you have to indicate first a name for the output TNT file, and subsequently select the tps files that are going to be included in that file. The file itself should then be ready to read into TNT. The command to merge tps files into a single file in TNT format is *dmerge*. The name of the species is extracted from the ID line. When landmark data is in 3D, the header of each individual should be LM3.

Example. Merge into the same tnt file the landmark data present in the files wings.tps and heads.tps.

```
dmerge | combined.tnt = wings.tps heads.tps ;
```

A species can be missing from some files, case in which it has only missing entries for that landmark. Make sure that all the species are named in exactly the same way in each tps datafile (otherwise, those will be considered as different species; optionally, TNT can be made to check for minor spelling differences in the different species, but this requires that the option [match N] is added to every

block, where N is the level of name similarity to consider two species as possibly the same; most spelling errors are in the order of N=0.9 to 0.95).

4.4.2. Outputting Results

4.4.2.1. Modified Matrices

If the matrix in memory is modified either by realignment (*lmreal*) or by rescaling (*lmark rescale*), it is possible to save it in a new file. This can be done either from the menus (with **Data/SaveData**). With commands, you need to first open a log (=output) file, and then write the matrix to that file. The commands used are *log* and *xread**.

Example. Save the modified matrix in the file *myfile.txt*.

```
time-; report-; sil=all ; log myfile.txt; sil-file; xread*; quote ., procl., ;  
log/;
```

(this insures that timing or other information is not written to the file, saves the data, and then closes the file).

4.4.2.2. Ancestral configurations

It is possible to export the ancestral configurations in TNT format. This is done in a two-step process. After setting *lmark showhtu*, and mapping the configurations (*lmark map*), the coordinates of the landmarks for all htu's (internal nodes) are printed in the text buffer. To save this information in a datafile you should save the text buffer with the subsequent command.

Example. Save landmark configurations for ancestral nodes in the file *mydatafile.tnt*

```
lmark showhtu ; lmark map ; log mydatafile.tnt ; svtxt; log/;
```

4.4.2.3. Exporting Configurations in tps Format.

To export coordinates from terminals (optionally ancestors as well) in tps format, use the following command:

```
export | filename C T
```

where C represents the configuration to be exported. If a tree (T) is specified, the tps file will also include ancestral configurations. If no tree is specified the tps file will include only configurations for terminal taxa.

Example. Export the 3rd landmark configuration (i.e. configuration 2) of terminals and ancestral nodes for the first tree (i.e. tree 0)

```
export | conf2_tree0.tps 2 0 ;
```

4.4.2.4. Drawings

It is possible to export drawings of ancestral configurations for any node in svg format using the command *lmbox*. This function cannot be called from the menus. The same options available for mapping configurations on a tree are also available here (e.g. include landmark numbers, connectors, wireframes, etc.). In the case of configurations in 3D, this command allows drawing different “views” of the configuration in the same *svg* file.

The usage is *lmbox filename T C = L ;* where T is the tree, C is the landmark configuration, and L is a list of terminal or internal nodes. The drawing options should be specified before the “=” (see the options with *help lmbox*).

Figure 1. Ancestral shape

Another way to save drawings of landmark optimization is saving the whole tree from the pre-visualization screen (**Optimize/Characters/MapLandmarks** and pressing “M” when the landmark optimization is shown on the tree). Except for small trees, this produces a drawing where each configuration is too small to appreciate the shape changes. A better alternative is to perform the same procedure but instead of saving the whole tree to a file, you can make a screen capture of the part of the tree of interest.

4.4.2.5. Export Tree with Branch Lengths

It is possible to export a tree file where branch lengths are proportional to the sum of landmark changes along the branches. This is done with the command *blength*. You can calculate the branch length for selected landmarks, selected configurations or complete datasets. Branch lengths are calculated according to the standardization factors that are in effect at the moment of optimizing landmark data for the last time. To calculate branch lengths for particular landmarks or

configurations you should deactivate all landmarks/configurations and then activate those you are interested in.

Example. Calculate the branch length for configuration 0 on the tree 0:

- 1-. *lmark wts* = 0 /. ; (give weight of 0 to all landmark configurations)
- 2-. *lmark wts* = 1 ./0 ; (give a weight of 1 to all landmarks from configuration 0) .
- 3-. *blength* 0 ; (calculate the branch length for tree 0 ;

In the previous example branch lengths are shown as a table. To export a tree file in nexus format including branch length after the first two steps

- 1-. Repeat the first two steps of the previous example.
- 2-. *ttag* - ; (**Trees/MultipleTags/ClearTags**) erase previous tree tags.
- 3-. *ttag* = ; (**Trees/MultipleTags/StoreTreeTags**) start storing tree tags.
- 4-. *blength* * 0 ; calculate branch length for tree 0
- 5-. *export* * > mylandblen.tre ; export the tree with branch lengths in nexus format.

It is also possible to use macro functions to generate graphics in svg format of the tree with branch lengths.

```
macro=; ttag-; ttag=; blen *N;
loop 0 nnodes[N]
  if ( #1 != root )
    set 0 $(0,0,0,2:$ttag #1); ttag <#1; ttag +#1 $0;
  end
stop ;
ttag;;
proc/;
```

Reemplazandole el "ttag:" del final por:

ttag & filename.svg color ;

There is a script (http://www.lillo.org.ar/phylogeny/tnt/scripts/land_brlen.run) that automatically calculates the branch length for the first tree in memory and produces one nexus tree file per configuration.

4.4.3. Missing Data

TNT can cope with missing data in the case of landmark optimization. Missing data may involve individual landmarks for a given configuration or complete configurations. The symbol to indicate missing data is "?" (in tps files,

the value 9999 for each coordinate is also read as a missing entry). One "?" symbol per landmark should be included (neither one symbol per coordinate nor one symbol per configuration).

```

xread
p 5
&[landmark 2d]
Prosopis_alba      ?      0.812,1.031 0.812,1.031
Prosopis_nigra     0.123,0.002 0.402,2.312 0.402,2.312
Prosopis_pallida    0.002,0.031 0.912,2.091 0.912,2.091
Prosopis_chilensis 0.034,0.112 0.345,1.531 0.345,1.531
Prosopis_caldenia  0.034,0.112 0.102,1.431 ?
&[landmark 3d]
Prosopis_alba      0.1234,0.02,0.098 0.402,2.312,0.39 0.402,2.312,0.920
Prosopis_pallida    0.0022,0.03112,0.2 0.912,2.091,0.03 0.912,2.091,0103
Prosopis_chilensis 0.034,0.1121,0.03 0.345,1.531,0.12 0.345,1.531,0.3
Prosopis_caldenia  0.345,1.531,0.3 0.034,0.1121,0.23 0.345,1.531,0.03
;
proc/;

```

Prosopis alba has a missing entry for the first landmark of the first configuration. *Prosopis caldenia* has missing entry for the third landmark of the first configuration. Notice that *Prosopis chilensis* is not in the second block. In this case TNT considers this species as missing for the second configuration (i.e. it is not necessary to include missing data for all the landmarks of the configuration).

TNT cannot perform dynamic superimposition using a tree as guide when some landmarks have missing entries. If the dataset includes several configurations (i.e. structures), only those with no missing data will be superimposed. The other superimposition approaches (RFTRA, pairwise-linear) accept missing data.

4.5. Algorithms to Optimize Landmark Data

The algorithms described in this section allow establishing the optimal ancestral shapes for a given tree and superimposition of configurations. Algorithms for landmark searches and superimposition on a tree are described in other sections. Settings for landmark optimization are defined with the command *lmark* and/or in **Settings/landmarks/GeneralOptions**

The command to perform the optimization and mapping is *lmark map* and *lmark lscore C/T* where C is the configuration and T is the tree; *lmark map* gives the tree scores and shows the optimization of the specified landmark configuration in

the specified tree. If no tree/configuration is indicated, all configurations on all trees are shown.

4.5.1. Optimization of observed states

In this procedure the states (positions) assigned to the internal nodes in an individual landmark must correspond to the positions present in the terminals. The transformation costs are the distances between each position (but in this case, all the costs can be precalculated prior to optimization, thus saving time) . Once the cost matrix is calculated, the ancestral position for each landmark is obtained by the Sankoff algorithm (Sankoff & Rosseau 1975). The main limitation of this procedure is that the positions that minimize the score may not (and usually do not) coincide with the observed positions.

4.5.2. Optimization using grids

In this approach additional positions can be added as possible ancestral conditions (instead of only observed positions as in the previous approach). This approach can be effectively considered as a discretization of the space, where the cells of a grid are the states conditions that can be assigned to internal nodes. The procedure first superimposes a grid in such a way that all the space occupied for the landmark of all terminals is fully covered. Then, a cost matrix is built including all the distances among all pairs of cells. Once the cost matrix is generated, the character is optimized by means of the Sankoff algorithm. The optimal cell is an approximation of the optimal position for the optimized landmark. The precision depends on the number of cells included in the grid: the higher the number of cells, the higher the precision. However, the execution times of Sankoff's algorithm increases quadratically with the number of states, making it necessary to find a value of compromise. In general, grids between 6x6 to 10x10 cells are a good compromise between precision and execution time.

Example. Optimize landmark data using a grid with 5 cells per side (=25 cells in 2D, 125 cells in 3D).

lmark cell 5 ;

4.5.3. Nested grids: Once an optimal cell is defined for each node, the grid approximation can be repeated considering a more reduced grid, centered around the optimal cell. This new grid can also include cells adjacent to the optimal cell.

Example. Use a grid of ten cells on each side (100 cells in 2D, 1000 cells in 3D), and conduct two levels of nesting, superimposing the grid on the best cell of the previous level and cells that are one cell apart from the optimal cell.

lmark cell 10 nest 2 1

4.5.4. Combining grids and observed states.

It is also possible (and the default) to run the Sankoff algorithm including as possible states the combination of states defined by the grid approach and the observed .

Example. Optimize configuration 3 on tree 0 combining grid and observed states. Use a grid of six cells per side, one level of nesting, with a window of one cell.

lmark termoints ;

lmark cell 6 nes 1 1 ;

lmark map 0/3 ;

4.5.5. Grid Shaking, Grid Shrinking, Cell skipping

The results of the grid approach can be affected by the particular position of the grid. Small movements of the grid may improve the scores. This is done with the option *lmark shake* N where N is the number of different “shakes” to be performed. The running time, obviously, increases linearly with the number of shakes.

The option *lmark shrink* allows increasing the precision (by diminishing the surface covered by the grid in cases when one (and only one) of the observed positions for a particular landmark is far away from the rest --no internal node can then receive such an outlier position in any optimal assignment. In general there is no important difference in time using this option. A similar result can be achieved by the option *lmark skip* where the program tries to identify and exclude cells that will not probably be optimal in certain nodes.

By default TNT uses two rounds of shaking, no skipping and no shrinking.

4.5.6. Node-by-node Iterative refinement

Once initial positions are assigned to all nodes of the tree using one of the procedures just described, it is possible to improve the score by modifying the positions node-by-node. The position of each landmark is modified in each internal node in order to minimize the difference in position of the landmark in the chosen node and its neighboring nodes (in a binary tree, ancestor and two descendants). For the positions of these surrounding nodes fixed, the coordinates for the middle node can be calculated exactly for bifurcations (with Torricelli's geometric construct for finding the Fermat point of a triangle), and with great level of precision for multifurcations. However, the positions for the surrounding nodes themselves may require changes after re-positioning the middle node, so that the procedure is heuristic, and needs to be performed iteratively (until a local optimum is found). In TNT, all the nodes are visited iteratively either until a user defined number of cycles is completed, or the improvement between two consecutive rounds of iteration is lower than a certain threshold. This approach is set in **Settings/Landmarks/GeneralOptions** option "*iteratively calculate Fermat/geometric medians*" (which is the default option). The commands to perform the iterative refinement are: *lmark iter* and *lmark maxiters N C*, where:

N = maximum number of cycles for node-by-node improvement.

C = value of score improvement between two cycles of improvement, below which the procedure is stopped. By default TNT uses the iterative improvement with a maximum of 100.000 iterations and 0.000001 of difference in score between iterations as cut value.

In practice, it is always advisable to use node-by-node improvement because the time required is small (relative to the time required by the Sankoff optimization of the initial grid), and the improvement is significant. Yet, providing very poor initial values (e.g. observed conditions only) shows that the iterative refinement can still be easily trapped in local optima, thus making it necessary to use an initial grid. Therefore, the default values include both the use of an initial grid and iterative refinement, which works best in most of the cases.

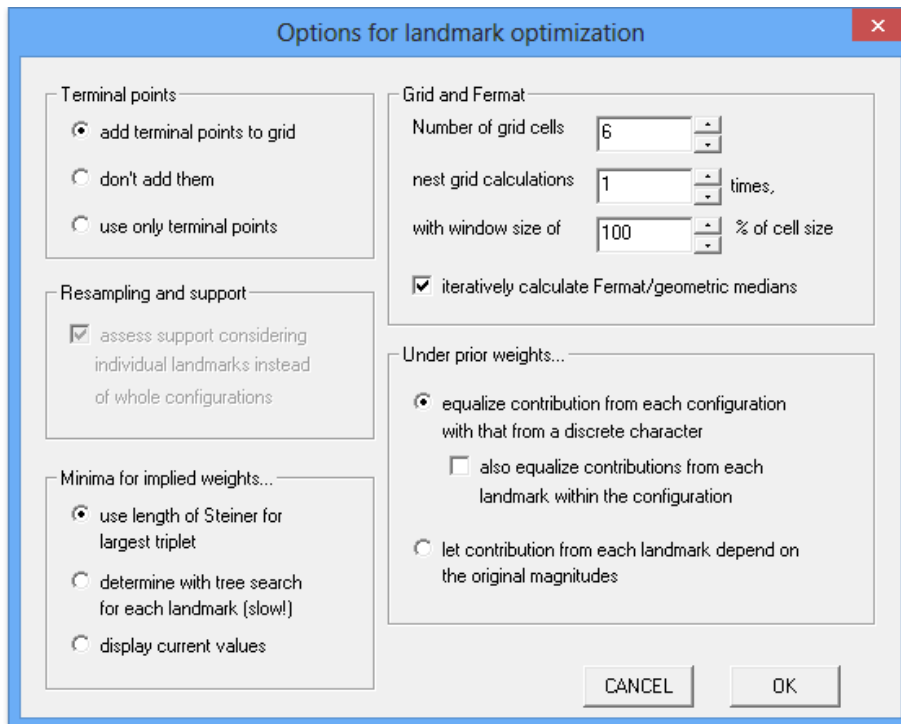


Figure 1. Windows Dialog for setting landmark optimization. Settings/Landmarks.../ GeneralOptions).

In the dialog for general settings of landmark optimization, the box “Terminal points” defines the settings for using the positions observed in the terminals for optimization. The option “add terminal points to grid” uses the combination of the grid approach with positions observed in terminals. The corresponding command is *lmark termpoints*. The option “use only terminal points” disables the grid approach. The corresponding command option is *lmark termpoints - (termpoints + reestablishes the grid)*. The “don’t add them” option of the dialog defines the use of the grid approach not adding terminal points. The corresponding command is *lmark notermpoints*.

Sometimes there are landmarks that present almost no change in position along all taxa. In order to save time it is possible to skip the optimization of those landmarks using the command *lmark maxprec N*. To do this, TNT calculates the maximum distance between terminals for each landmark. Those landmarks with a maximum distance below N are simply not optimized. By default every landmark is optimized (i.e. *lmark maxprec 0*).

4.6. WEIGHTING/SCALING/FACTORS/STANDARDIZATION

4.6.1. Weights

The weights defined with the *ccode* command (or with **Data/CharacterSettings**), used for traditional characters as well, will also affect landmark configurations; they will do so as a whole, however, and cannot modify the weight for individual landmarks.

The weight of individual landmarks can be changed with the option *lmark wts*.

The syntax is: *lmark wts* = *W C/L*.

This sets weight of landmark *L* of configuration *C* to *W*. If no *L* is specified, the weight *C* is given to all landmarks in configuration *C* (i.e. similar to weights for discrete characters, but with the possibility of using floating point precision for the weight). These weights are independent of the standardization factors (i.e. if automatic factors are on, the weights multiply the standardized scores). The default weight for each landmark is 1; to display current weights for each landmark in each configuration type *lmark wts* with no further arguments.

4.6.2. Factors

By default, for each configuration, TNT standardizes the contribution to the total score by calculating factors. This standardization is carried out in such a way that the sum of the maximum possible difference between two terminals for each landmark in the configuration costs as much as a step in a discrete character. This standardization does not modify the coordinate values themselves, but uses instead factors to multiply the scores for the individual landmarks. While the standardization is intended to affect tree searches (i.e. choice of trees), the shapes inferred for the ancestral nodes of a given tree are exactly the same. A complete explanation of how factors are calculated can be found in Catalano et al. (2010).

The command to define the standardization among configurations is *lmark fact* = * (which is the default option in TNT). To turn off the standardization, the command is *lmark fact* = 1, case in which no standardization is performed and the total score of each configuration is just the sum of absolute landmark displacements (possibly weighted by the weights as defined in the previous section).

It is also possible to calculate standardization factors in such a way that the contribution of all landmarks within a configuration is similar irrespective of the maximum displacement possible for each landmark. This standardization within each configuration is combined with the standardization among the configurations.

The command to perform standardization within the configurations is *lmark inscale* (*lmark noinscale* turns it off).

The use of factors to standardize the contribution of each configuration and/or landmark, as explained, above does not modify the data matrix: the scores are first calculated on the original data, and then multiplied by the standardization factors. It is also possible to obtain the same result (i.e. the same scores as after the standardization) if each coordinate in the original data is multiplied by the corresponding factor (and then resetting every factor to unity). The command to perform this is *lm rescale = **. This does modify the matrix stored in memory, and is important when using implied weighting or realignment (see below).

To display the factor for each landmark in each configuration type *lmark factor ;*. Note that the default values are those calculated by TNT on the basis of the original matrix, and thus change with different data sets.

Note: do not use *lm rescale* if the *inscale* option is on. This is because *inscale* determines a different factor for each landmark. If the configuration is rescaled with those factors, the configurations are literally deformed.

The menu option to define factors and scaling is **Settings/Landmarks/GeneralOptions (Fig x.)**. The box “*under prior weights*” determines the different ways to assess the contribution of individual landmarks and configurations (see point 4.6). “*Equalize the contribution from each configuration with that of a discrete character*” is equivalent to *lmark factor = ** and is the default option in TNT when the data is read. “*Equalize the contribution from each landmark within the configuration*” corresponds to the command *lmark inscale*. “*Let the contribution from each landmark depend on the original magnitudes*” corresponds to *lmark factor =1* . In all cases, these settings affect only the calculation of the scores but not the original coordinates (as in *lmark rescale*).

4.7. Mapping Shapes

4.7.1. Visualizing ancestral landmark configurations

The ancestral shapes can be visualized on the tree pre-viewing screen, in the Windows menu version of TNT. Make sure that the preview mode is ON

(**Format/PreviewTrees**). TNT shows one configuration (i.e. one character) at a time, for all the nodes of the tree. The command to map configurations is:

lmark map N/C

Where N is the tree and C the configuration.

If no tree is defined, TNT maps the specified configurations on every tree in memory. The same applies to configurations. To map configurations from the menus, select **Optimize/Characters/MapLandmarks**. Besides showing the ancestral shapes, the program also shows the score for each configuration mapped. Finally, the sum of the scores for all the configurations mapped is written to the text buffer.

Several features of the display for landmark mappings can be modified. First, the size of the displayed configurations can be modified with the “+” and “-” symbols. The size of the whole tree can be modified with F3-F4 (width) and F5-F6 (height). It is also possible to hide all the boxes (with “N”, for "nude"), then making only the node(s) of interest visible by left-clicking on those nodes; pressing “A” shows the boxes for all internal nodes, and “T” for all terminals.

In order to better visualize the changes occurring in a particular branch it is possible to superimpose the shape with that of its ancestral configuration. The ancestral configurations are displayed in gray dotted lines. This is turned off/on pressing “O”.

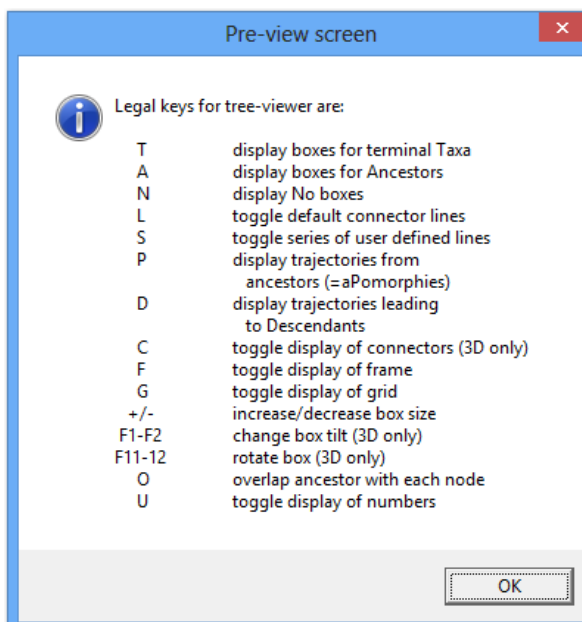


Figure 2. Help for pre-visualization screen (Accessible pressing “H” while showing landmark optimization).

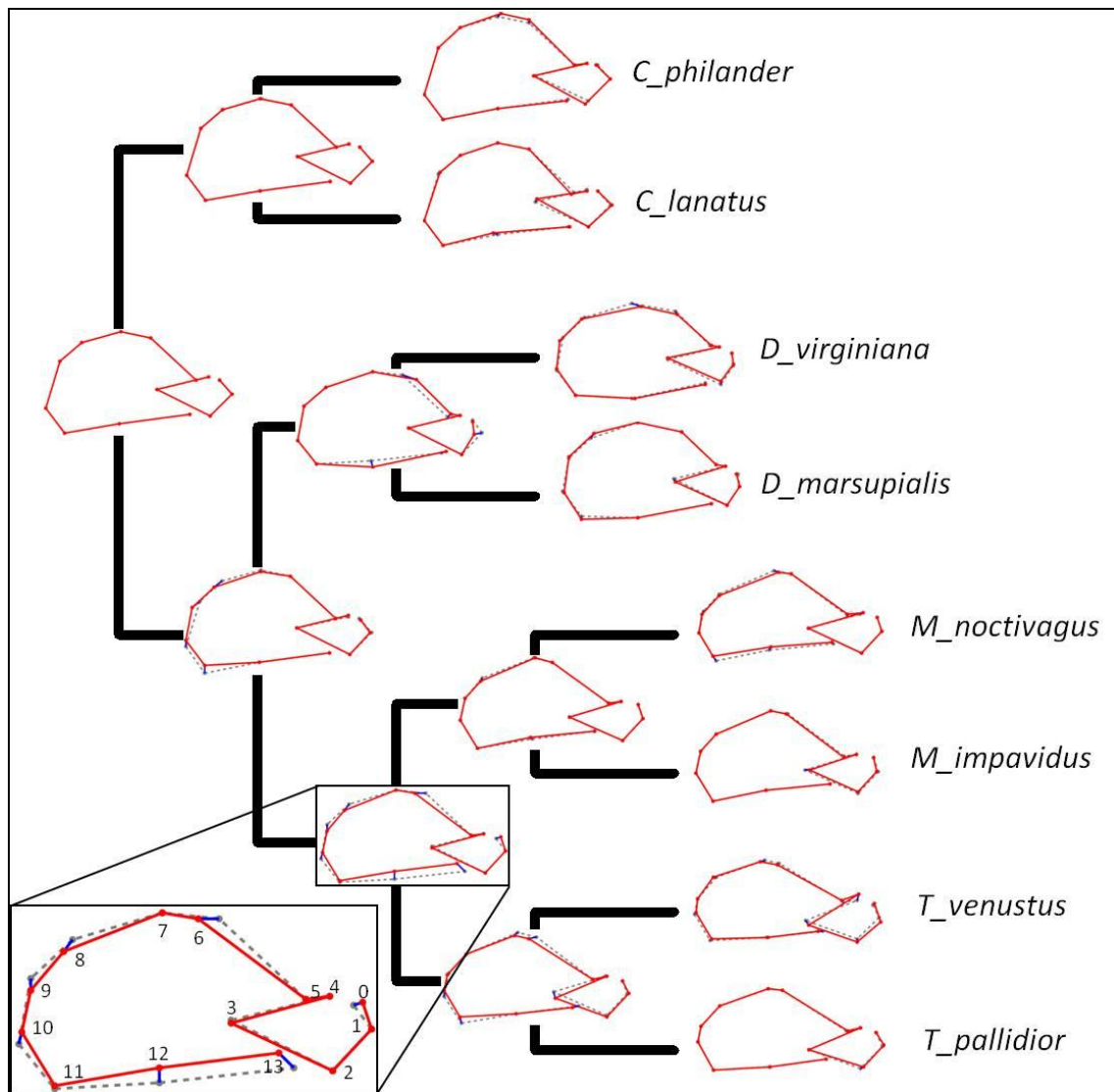


Figure 3. Screen capture of a landmark configuration optimization. In each node it is possible to superimpose the ancestral shape (grey-dotted line) with the shape inferred for the node (red lines and points). The landmark displacement inferred between ancestor descendant pair is shown in blue.

TNT allows including a wireframe connecting the landmarks of each configuration. By default, the wireframe follows the order of the landmarks in the dataset. The displaying of the wireframe is turned ON/OFF pressing “L” at the tree pre-visualization screen. Alternatively the user can define a different wireframe (i.e. user defined landmark connections). The best way to do this is to include it in the datafile itself, after the landmark data, using the command *lm connect*. The format is:

lmark connect C Li-Lj Lj-Lk / D Li-Lk Lj-Lk;

Where *C* and *D* are configurations, and *L-L* are the landmarks to connect. It is possible to define several lines departing from the same landmark. All the wireframes should be included in the same call to *lm connect*, as every call to this option overrides all previous wireframe definition. In the tree pre-visualization screen, the user defined wireframes are turned ON/OFF by pressing “S”.

Example. Draw a wireframe connecting landmarks 0 with 1, 1 with 2 and 3 with 4 of configuration 0 and another wireframe connecting landmark 0 with 2 and 0 with 3 of configuration 1:

lmark connect 0 0-1 1-2 3-4 / 1 0-2 2-3 ;

Additional options for visualizing landmark optimization:

- Configurations for ancestral nodes can be hidden / made visible by pressing “A”
- Configurations for terminal nodes can be hidden / made visible by pressing “T”
- Configurations for terminal nodes can be hidden / made visible by pressing “N”
- The number of each landmark can be shown pressing “U”.
- To visualize ancestor/descendant differences press “P”. The change (i.e. landmark displacement) is shown in blue. You can also visualize the change from the node to its descendant pressing “D”. In this case the changes are shown in green.

All the settings for visualization can also be modified using commands. Most of these settings are modified with the option *lmark line*. The usage is:

Lmark line +cgfplsd. Where “c” are lines connecting the base of the box to each landmark (3D only), “f” is a frame (or box in 3D), “p” are the movements in each landmark from the ancestor to the node, “l” is the default wireframe, “s” is the user defined wireframe and “d” are the changes in each landmark from the node to the descendant.

Example. Show only the box frame and user defined lines.

lmark -cgpld +fs ;

4.7.2. 3D visualization

You can rotate the view of 3D configurations using F1-F2 and F11-F12. In addition you can use “connectors” (Pressing “C”) to better visualize the configurations. The commands to rotate the view are *lmark rot N* and *lmark tilt N*, where N is expressed in degrees (i.e. 360 represents a full rotation).

4.8. Ambiguity in Ancestral Landmark Reconstruction

lmark multimap, lmark amb ;

Except for artificial examples, or in the case of missing data, it is very uncommon to have ambiguity in ancestral landmark positions. This is because it is very unlikely that different landmark position produce the same score (a score that is measure with a precision of many decimals). However, there are two options of the command *lmark* related to ambiguity. The option *lmark multimap* allows showing the ambiguity in landmark mapping. The program repeats the mapping several times, randomly choosing in every case one optimal position. The result is a cloud of positions for the given landmark in each node; a small cloud indicates that the landmark position can be determined with precision, and viceversa. The different positions can in fact have two different causes: (i) actual ambiguity, or (ii) errors associated to the heuristic nature of the algorithms. Hence, if the actual ambiguity is the only interest, the user may set strong optimization settings (e.g. *lmark cell 12 nes 4 iter*), so that most of the variation will be caused by real ambiguity. Alternatively, this can be used to easily explore the degree to which different parameters for the heuristic procedure of finding optimal coordinates produce different results.

The other option of *lmark* to deal with ambiguity is *lm ambig*

4.9. Score calculation

If the matrix contains landmark data and traditional characters the score given by *lm map* is the sum of the scores for all the active configurations, without summing the scores of traditional characters. The scores of landmark configurations plus traditional characters is given by the commands *score* and *length*; note, however, that *score* and *length* do not differentiate the contributions for each individual landmark (only *lmark* does).

To calculate the score for particular trees /characters/landmark the command is *lmark lscore T/C/L*, where *T* is the tree, *C* is the configuration and *L* is

the landmark. The macro expression that returns landmark scores is *lmscore* [T C L].

IMPORTANT ! Since landmark optimization uses heuristic approaches, the scores will depend on the particular settings defined for landmark optimization. Hence, you should not compare the optimality of alternative trees under different optimization settings.

4.10. Superimposing Configurations

After reading a landmark dataset, TNT will analyze the configurations as superimposed in the datafile. However, the multiple superimposition of the configurations can be modified in TNT. All the commands to realign the configurations modify the matrix in memory, so that subsequent commands are executed on the modified matrix. The original file is NOT modified by the alignment commands. Consequently, to keep the new multiple superimposition, the data in memory should be saved into a file with the command *xread** or with **Data/SaveData**.

There are four different approaches to superimpose configurations in TNT: linear pairwise, RFTRA, two-point registration and superimposition using a tree as guide.

TNT does not calculate GPA superimposition. If the user prefers to analyze the data using this superimposition, the original data should be already aligned with this procedure.

4.10.1. Pairwise superimposition.

This is a so-called ordinary alignment where all the configurations are superimposed against a reference taxon. Two different criteria can be used:

- Minimizing the sum of Euclidean distances between corresponding landmarks. The command is *lmreal pairlin*. This options is available for 2D and 3D configurations.

lmreal pairlin [N] /C , where N is the reference taxon and C is the configuration to be aligned.

- Using repeated medians *lmreal rftra* (RFTRA, Benson and Hedges 1984). (Available only for 2D configuration).

lmreal rftra [N] /C , where N is the reference taxon and C is the configuration to be aligned.

By default, *lmreal pairlin* and *lmreal rftra* do not modify the size of the configurations; they only rotate and translate the configurations. To also modify the size of the configurations the symbol “!” should be added: *lmreal rftra [C] !*;

4.10.2. Two point registration (2D)

TNT also allows superimposing the configurations in all the taxa, so that two chosen points are perfectly superimposed in all the configurations. This method is only available for 2d.

lmreal twopoint (L₁ L₂) /C where L₁ and L₂ are the two landmarks considered as reference and C is the configuration to be superimposed.

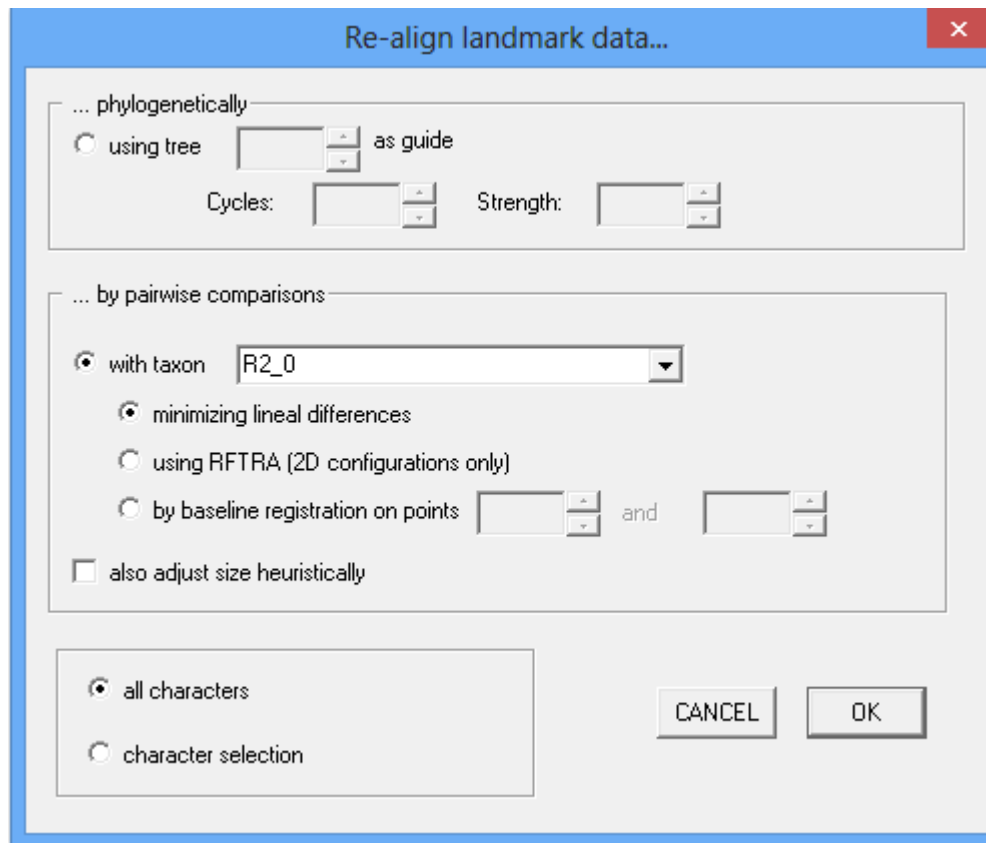
4.10.3. Using a tree as guide (2D-3D)

TNT implements the ideas presented in Catalano & Goloboff (2012) to superimpose landmark configurations taking a tree as a guide. The method is a heuristic that seeks a multiple alignment to minimize the sum of landmark displacements along the tree. For details see Catalano & Goloboff (2012).

Usage:

lmreal tree T level L cycles Y / C

Superimposes configuration C on tree T using a level of thoroughness of L and performing Y cycles of improvement/perturbation. “level” (0-4) defines the thoroughness of the alignment improvement at every cycle. A level of 2 is generally a good compromise between score improvement and time. The option “cycles” defines the number of improvement/perturbation cycles. The perturbations imply small random changes in the position and rotation of the configurations followed by reoptimization, to attempt to escape from locally optimal alignments.



In the Dialog for re-aligning landmark data, the box “...*phylogenetically*” allows running the superimposition considering a tree as a guide. The corresponding command is *lmreal tree* and “strength” is equal to the option *level* of *lmreal tree*.

The box “...*by pairwise comparison*” includes all the approaches that superimpose every configurations against a single reference taxon. The corresponding command to run pairwise comparison by minimizing linear distances is *lmreal pairlin*. The command to run RFTRA superimposition is *lmreal rftra*. The command to run twopoint registration is *lmreal baseline*. To adjust size in RFTRA and pairwise linear superimposition the symbol “!” should be added.

Keep in mind!

There are some circumstances on which after you modify your alignment you cannot compare the scores of one or more trees. This happens when:

- You modify the size of the configurations
- You use automatic standardization (see *lmark inscale* /*lmark rescale*).

In the first case you cannot compare the scores because these are a function of the difference in the positions of the landmark. If the scale is modified, the score is also modified without implying any real improvement.

In the second case, the standardization factors are based on the distances among the observed positions of each landmark. When the alignment is modified, these distances are also modified and consequently the factors used for producing the final scores are different. This problem can be circumvented if the standardization factors are used to rescale the configurations held in memory using the command *lm rescale = **. Once this is done, if the factor is set to unity (i.e. no further automatic standardization factor is used), the scores before and after the superimposition are comparable.

4.10.4. Realignment and phylogenetic searches (see Searches section)

4.11. Phylogenetic Searches

4.11.1. Generalities

Settings for phylogenetic searches including landmark data are defined in **Settings/Landmarks/BranchSwapping**.

One of the most important differences between phylogenetic morphometrics and other approaches to infer phylogenetic relationships from landmark data is that only the former is completely compatible with the use of different sources of evidence in a single analysis. Thus, TNT allows combining DNA, discrete, continuous and landmark data (see standardization section). All the tree search algorithms implemented in TNT are transparently available for matrices that include landmark data. In real datasets it is quite rare that more than a single optimal tree is obtained from the analysis of landmark configurations (because exact ties in tree scores are very unlikely). A counterpart of this is that trees with a minimum difference (e.g. +0.0001 in score) will not be considered as optimal. As a consequence of this it is VERY IMPORTANT to calculate support measures to determine which groups are supported and which groups are weakly supported. Otherwise, the tree will be over-resolved.

4.11.2. Searches Considering Multiple Configurations

When phylogenetic searches are conducted considering multiple configurations, a key factor is how to sum up the score of the different configurations. TNT automatically calculates a standardization factor that is taken into account when the combined score for all the configurations is calculated. Hence if you read your datamatrix and perform a phylogenetic search without changing any setting you are using this standardization. If you want that the score calculated for each configuration is given by the sum of landmarks displacements for all landmarks within a configuration you have to set *lmark factor* = 1. Note that this will produce that configurations in larger scales and/or higher number of landmarks will have more preponderance in the analysis.

4.11.3. Group Support

Group support considering landmark data alone or in combination with other characters can be assessed by means of Bremer support and/or resampling procedures. In the case of absolute Bremer support, the calculations are done just as for standard characters; no special precautions are needed. In the case of resampling, there are two options: (i) resampling individual landmarks from all the configurations (ii) resampling configurations as a whole. The latter makes sense only when you have several configurations (8-10 or more), or when you combine a few landmark configurations with traditional characters.

If the dataset only includes a single or a few configurations, resampling landmarks within configurations is the only logical option. This also makes sense taking into account that TNT treats the “configuration” as the “character”, but different individual landmarks within a configuration may define conflicting groups (something precluded for standard characters). The resampling will then evaluate the conflict between these landmarks.

The option to choose resampling and relative Bremer support considering either whole configurations or individual landmarks is defined in the “*resampling and support*” Box (selected in the menus with **Settings/Landmarks/GeneralOptions**). The corresponding command is *lmark confsample* (or *lmark noconfsample*). The default option is to assess support by considering entire configurations.

4.11.4. Searches and Realignment

In theory, the correct way to perform phylogenetic searches would be by calculating the score considering the best alignment on each tree. This would

require performing a different realignment on every possible tree (or every tree generated during branch-swapping). This is highly impractical for datasets of 20-25 taxa or more, due to the time required. A possible approximation is to perform cycles of tree search /re-alignments until the tree obtained does not change in two consecutive cycles. This has the problem that the search can be trapped into a local optimum depending on the original alignment. To circumvent this problem the following strategy can be followed:

- 1-. Start several replications (Wagner trees + TBR), each searching from a different superimpositions (for instance superimposing all taxa against a different reference taxon (without changing the size!!)
- 2-. Perform a phylogenetic search.
- 3-. Once a tree is obtained superimpose the configurations over this tree (*lmreal tree 0*);
- 4-. Repeat 2 and 3 until the tree obtained is the same as the one used as guide.
- 5-. From all replications, keep the tree with the lowest score.

Remember that in order to make scores comparable along the whole process the configurations should be standardized prior to the analysis (with *lm rescale = **) and the standardization factor should be set to one (*lm fact=1*).

4.11.5. Implied Weighting

Implied Weighting (IW) is a parsimony method that weights characters according to the homoplasy observed for each character (Goloboff 1993). Landmark data can be analyzed with implied weighting in TNT. The homoplasy is calculated as the difference between the observed and the minimum possible displacements of the landmarks, and therefore the possible minimum for each character must be calculated. In the case of landmark data, this minimum can be set either by a phylogenetic search (landmark-by-landmark) or by a quick approximation to that value. This is set with **Settings/Landmarks/GeneralOptions**, in the box “Minima for implied weighting”. The default option in TNT is to approximate the minima using the length of the largest Steiner tree for any three landmarks –this is likely to *underestimate* the minimum, thus producing more apparent homoplasy. Accurate calculations of homoplasy can be obtained when

the minima for each landmark are found with a search for each landmark. This is set with “*determine with tree search for each landmark*”. The command to handle minima for implied weighting is *lmark usmin*. Once the minima are calculated in such way, subsequent analyses under implied weighting will use these values. Since calculating the minima in this way may require significant search time, the values calculated can be saved to a textfile (for subsequent input to the program).

This requires different steps:

- 1-. Calculate the minima, landmark by landmark, using a search as indicated above.
- 2-. Show the values obtained with **Settings/Landmarks/GeneralOptions** and the option “*display current values*” in the box “*Minima for implied weighting*”.
- 3-. Save the text buffer (with the *log* and *svtxt* commands).
- 4-. Copy the minima and paste it in the original data file following the datamatrix (and before the *proc;* !!!) . The dataset will look something like this:

Example.

```
xread
1 5
& [landmark 2d]
SP1 0.202563,-0.094253 0.127129,-0.118399 -0.288333,0.184120
SP2 0.248025,-0.128456 0.181367,-0.138127 -0.325229,0.134915
SP3 0.177217,-0.120636 0.093606,-0.132157 -0.325386,0.173186
SP4 0.044979,-0.129192 0.019846,-0.135383 -0.339789,0.049257
SP5 0.090376,-0.121996 0.012579,-0.124877 -0.329125,0.077029
;
lmark usmin
=0.082 0/0 =0.063 0/1 =0.062 0/2 ;
proc/;
```

There are two different options to analyze landmark data under implied weighting, and the choice of which one to use may depend on what the landmarks represent (or how the configurations are defined). One of these options is to weight each individual landmark according to its homoplasy; the second is to weight each landmark based on the average homoplasy of the configuration. By default, when landmark data are analyzed using IW, every landmark is weighted according to its own homoplasy. To weight an entire configuration (or several configurations

combined) by the total homoplasy of the component landmarks, it is necessary to use extended implied weighting (Goloboff 2014), defining a weighting set for the landmark(s) in question.

If IW is activated, the automatic factors to standardize configurations are deactivated (this is because the weighting formula takes into account the absolute value of the homoplasy). Consequently it is wise to rescale the configurations before the IW analysis, so the configurations themselves are already standardized in memory when IW is executed. The commands to do so are best included in the data file itself, thus avoiding the need to set this option every time you read the data file.

The commands to set weighting for entire configurations is `piwe=`; `xpiwe=`; `xpiwe [0]`; . Weighting for individual landmarks (the default) is set with `xpiwe-`;

The steps to perform a phylogenetic analysis under implied weighting for a dataset with multiple configurations are:

- 1-. Set implied weighting ON before reading the dataset (`piwe=` or **Setting/ImpliedWeighting/BasicSettings**)
- 2-. Read the dataset.
- 3-. Modify the configurations in memory so that those are standardized considering the automatic factors (`lm rescale = *`) .
- 4-. (Optional) Calculate the minimum score for each landmark with a landmark by landmark search. (`lm usmin = !`)
- 5-. To run IW by weighting landmarks individually according to its homoplasy nothing else should be done, just use either traditional searches (`mult` or **Analyze/TraditionalSearch**) or New Technology Searches (`xmult` or **Analyze/NewTechnologySearch**). To run IW weighting each configuration by the total homoplasy of its component landmarks, extended implied weighting should be turned ON before running the search:

`xpiwe=` ; (Activate extended implied weighting)

`xpiwe [0` ; `xpiwe [1` ; `xpiwe [2,` ; etc. (Define each configuration as a different set for extended implied weighting).

4.11.6. Fine Tuning of Searching Algorithms for Landmark Data

As indicated above, searches for landmark data use the same standard routines used for other types of data (Wagner trees, SPR,TBR, etc.). TNT uses several types of shortcuts and approximations to speed up score calculations during tree searches, described in detail by Goloboff & Catalano (2016). The corresponding settings can be changed with **Settings/Landmarks/BranchSwapping** or with the options *lmark refine*, *lmark errmarg*, *lmark quickwag* and *lmark precision*. Do not change these settings unless you are an advanced user of TNT and know what you are doing.

4.12. Calculation of Phylogenetic Signal

Klingenberg & Gidaszewski (2010) proposed that the significance of phylogenetic signal in landmark data can be established by a permutation test. Although this approach can hardly be considered as a proper method to evaluate phylogenetic signal as claimed by the authors (“similar species tend to be grouped more closely than expected by chance”), it is still useful to have a certain idea of the agreement of the phylogenetic information of landmark data and alternative sources of evidence (i.e. the reference topology). There is a script to perform such test at: http://www.lillo.org.ar/phylogeny/tnt/scripts/land_signal.run

4.13. Landmark Expressions in TNT Scripting

lmdims [C]

dimensions of landmark character (if not a landmark, then return 0 [discrete] or 1 [continuous])

lmerror [N]

return status of landmark approximation in last SPR/TBR search (if no search effected, values undefined). N indicates statistic to return: 0 mean; 1 stddev; 2 minimum error; 3 maximum error; 4 number of cases. If no N indicated, returns current error margin

lmfactor [C L]

factor for character C, landmark L

lmmaxdist [C L]

maximum point distance, char C , landmark L

lmnchar

total number of landmark characters minus 1

lmnpoints [C]

number of points (landmarks) in character C minus 1

lmscore [T C L]

score of landmarks for tree T (optional, ch. C, landmark L)

lmwt [C L]

weight for character C, landmark L

lmxcell [C L]

return x, y, or z dimension of cells. Expressions lmycell and lmzcell are equivalent.

lmxcoord [C L T N]

return x, y, or z coordinate for node N, tree T, landmark L, char C. This uses the value in the last optimization; be careful with this. Expressions lmycoord and lmzcoord are equivalent.

5. REFERENCES

Catalano SA, Goloboff PA, Giannini NP. 2010. Phylogenetic morphometrics (I):

the use of landmark data in a phylogenetic framework. *Cladistics*, 26: 539–549.

Catalano SA, Ercoli M, Prevosti F. 2015. The More, the Better: The Use of

Multiple Landmark Configurations to Solve the Phylogenetic Relationships in

Musteloids. *Systematic Biology* 64: 294-306.

Catalano SA, Torres A. 2017. Phylogenetic inference based on landmark data in 41

empirical datasets. *Zoologica Scripta*. 46:1-11.

Catalano SA, Goloboff PA. 2012. Simultaneously mapping and superimposing

landmark configurations with parsimony as optimality criterion. *Systematic Biology*.

61: 392-400.

Farris J. 1970. Methods for computing Wagner trees. *Systematic Zoology* 19: 83–92.

Goloboff PA. 2014. Extended implied weighting. *Cladistics* 30:, 260-272.

Goloboff PA, Catalano SA. 2011. Phylogenetic morphometrics (II): algorithms for landmark optimization. *Cladistics*. 27: 42–51.

Goloboff PA, Catalano SA. 2016. TNT version 1.5, including a full implementation of phylogenetic morphometrics. *Cladistics*. 32:221-237.

Klingenberg C.P., Gidaszewski N.A. 2010. Testing and quantifying phylogenetic signals and homoplasy in morphometric data. *Systematic Biology*. 59:245–2616.

LINKS

TNT homepage

<http://www.lillo.org.ar/phylogeny/tnt/>

Some R functions that are useful for subsequent analysis of landmark data in TNT:

- *Export continuous characters and landmark data from R to TNT format: (Eduardo Ascurranz)*

<https://github.com/eascarrunz/tnttools>

- *Calculate consensus configurations for every species and generate a TNT datafile.*

- *Visual Quick Tutorial (ppt).*

<http://www.lillo.org.ar/phylogeny/tnt/files/QuickTutorial.zip>

- *Scripting Documentation*

http://www.lillo.org.ar/phylogeny/tnt/scripts/General_Documentation.pdf