TNT (Tree analysis using New Technology)

Introduction

TNT is a parsimony program by Pablo Goloboff, Steve Farris, and Kevin Nixon that can:

- analyse large data sets (i.e. 300-500 taxa)
- in reasonable times (minutes to find a shortest tree, hours to produce a reliable consensus).

The program is described in Goloboff PA, Farris JS, Nixon KC (2008). TNT, a free program for phylogenetic analysis. Cladistics 24:774–786.

The program is often used in the Cladistics community.

Data file format

TNT reads matrices in Hennig86 format, with some refinements. Normally, the data will be included in a file. Character states may be IUPAC codes, digits (for morphological characters), ? (for missing data), or - (for gaps). Simple Nexus files can also be used with variable success.

The basic format for data input is:

```
xread 'optional title, starting and ending with quotes (ASCII 39)'
nchar ntax
Taxon0 0000000000
Taxon1 0010111000
Taxon2 1011110000
Taxon3 1111111000
.....
TaxonN 1111111000
; '<- Note the semicolon (and the way you add comments)!'
```

`Seqret` program in EMBOSS can be used for formatting.

The data can be defined as DNA using the command `nstates dna;` or proteins `nstates prot;` in which case the IUPAC codes (including polymorphisms) are recognized and used throughout. For 32 states, non-protein data, the symbols 0-9 and A-V indicate states 0-31.

Commands

There are 150 different commands in TNT, which can be see with the `help` command:

absincl	agroup	alltrees	ancstates	apo	**bbreak**	beep
best	bground	blength	blocks	break	bsupport	ccode
cdir	change	chkmoves	chomo	ckeep	cls	clbuffer
cnames	collapse	comcomp	condense	constrain	costs	cscores
cstree	dcomp	dmerge	drift	edit	echo	export
fit	fillsank	force	freqdifs	help	hold	hybrid
ienum	incltax	info	lintrees	**log**	keep	kleex
length	lmark	lmbox	lmrealign	lquote	majority	matchtax
map	minmax	mixtrees	mono	mrp	**mult**	mxram
mxproc	naked	nelsen	nstates	outgroup	**procedure**	pause
pcrprune	pfijo	piwe	pruncom	prunmajor	prunnelsen	pruntax
prupdn	ptnt	qcollapse	qnelsen	quote	**quit**	randtrees
ratchet	rcompl	rdir	rebuild	recons	report	reroot
resample	resols	rfreqs	riddup	rseed	run	save
screen	scores	**sectsch**	shortread	shpcomp	silent	slfwt
slaveproc	smatrix	sort	sprdiff	subopt	svtxt	system
tables	tagset	taxcode	taxlabels	taxonomy	taxname	tchoose
tcomp	tequal	tfuse	tgroup	thanks	timeout	tnodes
tplot	tread	**tsave**	tshrink	tsize	ttags	txtsize
tzert	view	vversion	warn	watch	xcomp	xgroup
xinact	**xmult**	xperm	xpiwe	xread	xwipe	unique
unshared	usminmax	zzz				

Preliminaries

First, increase TNT memory usage to 1 GB or whatever is realistic and necessary: `mxram 1000;`. The default -- 16.78 Mbytes -- is too low for most tasks.

Second, enter `nstates DNA` or `nstates protein`, respectively, if using sequence data. Enter `nstates NOGAPS` to treat gaps as missing data as opposed to an additional state.

Thrid, use `taxname = ;` to use taxon names for terminal nodes.

Finally `log [logfilename];` will keep the log of your analysis.

Basic heuristic analysis with `mult`

For relatively messy, but not very big data sets, the best algorithm consists of multiple random addition sequences plus TBR (RAS+TBR); this is invoked with the `mult` command. Further branch-swapping starting from the trees in memory ("RAM") can be used to find the additional trees.

`procedure [filename];` or `p` will read the file;
`hold 100000; mult;` will do a basic analysis consisting of 10 random addition of sequences followed by branch-swapping with TBR, saving up to 10 trees per replication.
`bbreak=tbr;`

The info about the program can be printed with `help mult;`, while parameters can be seen with `mult:;` and changed with `mult:options;`

Once calculated, trees may be viewed by entering `tplot`. To save the trees for later reanalysis, create a file by entering `tsave * [tree_filename]`.

Here is an example of a TNT search using mult command:

```
hold 100000 ;
log tnt_run_log ;
taxname = ;
mult=replic 100 tbr;
bbreak = tbr ;
nelsen * ;
tchoose { strict } ;
ttags = ;  'store tree tags for subsequent tree printing command(s)'
blength *;
ttags );  'stop storing tags (but don't erase them)'
export - consensus_tree.tre ; 'export > consensus_tree.tre '
ttags -;  'clear all existing tags'
ttags = ;
collapse tbr ;
resample replications 100 ;
keep 1 ;
ttags );
export - boots.tre proc/;
```

More complex search with `xmult`

There are four basic types of special algorithms implemented in TNT: ratchet, drifting, sectorial searches, and fusing.

The ratchet (`ratchet`) consists of two phases, perturbation and search, which are sequentially repeated. The ratchet always uses TBR branch swapping, and it alternates the search phase (S) with three types of perturbation phase: original weights (O), upweighting (U), and deleting (D).

Tree-drifting (`drift`) is similar to the ratchet, but the perturbation phase, instead of reweighting the characters, accepts the moves with a probability that depends on both the relative fit difference and the absolute fit difference between the tree being swapped and the new tree. The perturbation phase also alternates between acceptance of optimal trees only (O), and suboptimal trees (U): O,S,U,S,O,S,U… etc.

Sectorial searches (`sectsch`) take a portion of the tree, create a reduced data set, and produce a mini-analysis of that data set. If a better solution for that portion of the tree is found, it is replaced back into the original tree. Every certain number of (user-determined) rearrangements, a round of global TBR is done (to insure global optimality).

Tree-fusing (`tfuse`) mixes trees, producing better trees. If the trees comes from different searches, the score is often considerably improved in very short time.

These algorithms are implemented in the `xmult` command. The default for the command are:

- Using 4 replications as starting point for each hit
- Each replication initially autoconstrained (previous and wagner)
- Each replication with constraint and random sectorial searches, with no ratchet, with drifting (5 iters.), no hybridization, and fusing (1 rounds)
- Finding best score 1 times (=hits)
- Not consensing trees during search
- Multiplying trees by fusing after hitting best score
- Saving no more than 1 trees per replication

One way to modify: `xmult=hits 10 noupdate nocss replic 10 ratchet 10 fuse 1 drift 5 hold 100 noautoconst keepall;`

Here is an example of using the xmult command in a search:

```
log tnt_run_log ;
mxram 2000;
nstates NOGAPS;
nstates dna;
hold 100000 ;
taxname = ;
'-----------'
xmult=hits 10 noupdate nocss replic 10 ratchet 10 fuse 1 drift 5 hold 100 noautoconst keepall;
bbreak = tbr ;
nelsen * ;
tchoose { strict } ;
ttags = ;
blength *;
ttags );
ttags ;
export - consensus_tree.tre ;
ttags -;
'-----------'
ttags = ;
resample replications 100 [ mult=replic 10 tbr hold 1000 ];
keep 1 ;
ttags );
ttags ;
export - boots.tre;
proc/;
```

A careful consideration of how the data set behaves to changes in parameters is the best way to proceed, but if you have no idea of how a data set behaves, or don't know how to set the parameters for a search, you may trust to the program the choice of parameters. This is done with the `level` option of the `xmult` command). The level must be a number between 0 and 10 (0 is very superficial; 10 is extremely exhaustive). If using a driven search (i.e. more than a single hit of the xmult command), you may have the program check whether best score is being found easily or not (and then decrease or increase the search level accordingly). This is set with the `chklevel` option of the `xmult` command.

Measures of character support

If there are multiple trees, their strict consensus can be found by entering `nelsen`. Resampling (jackknifing,

bootstrapping, etc.) can be done by entering `resample.`
Now download the file cox1*nt.tnt to your working directory and try this command:*
`*tnt p cox1*nt.tnt, echo=, log cox1_nt.out, rep+1, mu10=ho3, le, ne, resample, quit,`

What did you do?