POY version 3.0 documentation and command summary

update November 6, 2002

Phylogeny Reconstruction via Direct Optimization of DNA and other data

by

Ward Wheeler, David Gladstein, and Jan De Laet

Documentation by Daniel Janies and Ward Wheeler

Division of Invertebrates
American Museum of Natural History
Central Park West @ 79th St.
New York, NY 10024-5192

ftp://ftp.amnh.org/pub/molecular/poy

Quickstart

POY is a program that implements the direct optimization, fixed states, and search based alignment procedures of Wheeler (1996, 1999) as well as other character types. Although based on parsimony, POY can perform likelihood procedures.

The following are available at:

ftp://ftp.amnh.org/pub/molecular/poy

Binaries for the various platforms:

poy.exe
poy.linux
poy.osx

you can change the name of the binary to poy after downloading; make sure it is user executable on Unix.

the usage is:

```
poy -molecularmatrix matrixfile datafile -command > outputfile 2> errfile
```

standard error reporting (>2) now works under DOS for Windows NT and XP via the POY GUI (poy.tcl)

Source code and make files for POY are also available, up to date
PVM and OCAML and appropriate make files are required to compile these:

poysource.tar

```
gcc, PVM, and OCaml are required to compile these.
gcc is available from    ftp://ftp.gnu.org/gnu/gcc
PVM is available from    ftp://netlib2.cs.utk.edu/pvm3
OCaml is available from  ftp://ftp.inria.fr/lang/caml-light/
```

Hardware references:
Sterling, T. , Salmon, J., Becker, D. and Savarese, D. 1999. How to Build a Beowulf. A guide to the implementation and application of PC Clusters. MIT Press.
ISBN 0-262-69218-X

```
http://beowulf.gsfc.nasa.gov/beowulf.html
http://www.cacr.caltech.edu/research/beowulf/index.html
http://www.cacr.caltech.edu/research/beowulf/tutorial/tutorial.html
http://www.epm.ornl.gov/pvm/pvm_home.html
```

Sample datasets available:

`ftp://ftp.amnh.org/pub/molecular/poy`

*.flat are DNA data in POY block format.

*.ss is a morphological dataset in HENNIG86 (Farris 1988) format.

script.bat is a DOS batch file for one DNA + morphology analysis.

script is a simple UNIX shell script for DNA + morphology analysis.

*.txt are Sankoff or "step" matrices for costs of transversions, transitions, and indels among DNA sequences. See **`-molecularmatrix`** command.

POY trees in standard output are presented as taxa in nested parentheses followed by a treelength in brackets. With a little text editing, such POY trees can be viewed with various programs. See examples files produced from the test dataset of how to edit POY output. Alternatively, some commands produce files with parenthetical trees that can be directly imported in other programs (`-phastwincladfile, -jackwincladfile`) or with text-based graphical representation of trees (`-printtree`).

*.out is POY output

*.tv is a TREEVIEW file

*.tre is a WINCLADA file (one must open the *.ss file first, then the *.tre)

*.nex is a PAUP/MACLADE file (treeblock only)

TREEVIEW is available from:
`http://www.taxonomy.gla.ac.uk/rod/treeview.html`

WINCLADA is available from: `http://www.cladistics.com`

PAUP/MACCLADE are available from: `http://www.sinauer.com`

JACK2HEN.EXE (source jack2hen.c) is used to convert POY output (one or more trees in nested parentheses format) into a matrix (in HENNIG86 format) of group inclusion characters representing the consensus tree used for constraint files. JACK2HEN converts POY output into a consensus tree of n% as in 100 for strict consensus and lower numbers for more catholic structures. Strict and majority rule consensus trees can also be obtained directly (command -`printtree` and plotting options).

```
jack2hen n < poy_output_file > tree_outputfile
```

Typographic conventions in this manual:
1) command names in descriptions are bold, arguments and filenames are in plain italics, and the defaults and countercommands are specified inside brackets e.g.,

-commandname *argument* or *filename* [default countercommand]

multimode commands are explained as follows

-commandname *mode* [default mode]

```
mode 1: detail
mode 2: detail
mode 3: detail
```

2) command line, program input or output are in plain courier e.g.,

```
poy datafile1 datafile2 -commandA -commandB n -commandC filename > output
```

3) specific software and trademarks are referred to in UPPERCASE not bolded

Text control:
If input data, parameters, and/or constraint files are created or modified with word processors, it is crucial to make sure that the resulting files are saved as text only (or DOS text only). Some word processing programs, regardless of platform, may insert illegal, misplaced, and sometimes invisible characters that do not conform to formats required by POY. If such characters are present, POY will most likely exit with a cryptic error message. These problems can be avoided by using POY legal editors (but see also command -enabletmpfiles). BBEDIT [ftp://ftp.barebones.com] is a good Mac editor suited to UNIX. DOS users find that EMACS [http://www.gnu.org/software/emacs/emacs.html] works well. Word processors that show all characters (e.g., "show paragraph" command in MSWORD or "show codes" command in WORDPERFECT) might be useful to root out formatting errors.

Analysis of multiple loci:
Raw sequences can be optimized in POY. However CPU time and memory requirements will increase quadradically with the length of sequences (discussed in Phillips et al., 2000 but see also -iterativepass and –iterativelowmem). Thus cutting the sequences into fragments can save dramatically on computation time and space. Although this is not a necessity, standard practice for POY users is to delineate regions as they flanked by primer pairs, or by secondary structural features, or separate loci; treating each as a character in POY. In addition to speeding up the calculations, this practice allows for more specific user-controlled hypotheses of primary/putative homology between (parts of) sequences to be expressed.

Data can be imported into your editor of choice to be prepared for direct optimization in POY software as follows; for example:
http://jwbrown.mbio.ncsu.edu/BioEdit/bioedit.html
http://evolve.zoo.ox.ac.uk/software/Se-Al/main.html
ftp://megasun.bch.umontreal.ca/pub/gde

Data file formats:

The first taxon in the first data file is the default outgroup for the first replicate when `-oneasis` is specified or for all replicates if `-norandomizeoutgroup` is specified (see command `–outgroup` for overriding the default outgroup). The ordering of taxa need not be the same in all files although this is easier for editing files.

Genbank

```
mushroom
   1 ACGATAGC

frog
   1 AGGTTTAGAG

chicken
   1 TUTUTUTUU

cicadia
  (for missing data use two returns)
```

Block:

```
mushroom
ACGATAGC

frog
AGGTTTAGAG

chicken
TUTUTUTUU

cicadia
(for missing data use two returns)
```

HENNIG86

```
xread
''
5 4
mushroom    00000
frog        10000
chicken     11000
cicadia     11111
;
cc - 0.4;
proc/;
```

NOTE: POY can be finicky with this format, there must be spaces after "cc" and "-" and scopes must be explicit (e.g. "0.4" not "."). See also HENNIG86 docs at:
`http://www.gwu.edu/~clade/faculty/lipscomb/web.pdf`

Use:

All POY commands are preceded by a dash (such as branch swapping: `-tbr`). This could be commandA in the above command line. If a POY command requires an argument, that argument immediately follows the command (such as `-gap 2`). This could be commandB in the above command line. If a POY commands require a file this filename immediately follows the command (such as `-constrain` *`filename`*). This could be commandC in the above command line.

Data file names are not preceded by a dash and may be one of the three formats discussed above: HENNIG86, a modified Genbank input, and a version of "block" or interleave format (see examples of the manual). Character data, such as morphology, for which putative homologies have been established are entered in a HENNIG86 type format with xread command through weighting and additivity commands at the end of the file. Nucleic acid sequence data (including IUPAC ambiguity codes) are entered as either "block" files or Genbank files. The block files are simply pairs of sequence names followed by a simple unbroken string of bases and a carriage return. Do not include any gaps or dashes. Prealigned data can be used if all sequences are the same length. If the sequence data for a taxon are missing, the sequence name must be followed by two carriage returns. The Genbank format (like that of MALIGN [Wheeler and Gladstein, 1994-2000]), consists of the sequence name followed by a carriage return, and several lines of sequences carriage return at the end of each) consisting of first a number field (ignored) and several block of bases. Standard Genbank has five or six blocks of ten bases, but POY is more accepting so the blocks may vary in number and length. At the end of each sequence an additional carriage return separates the sequence form the next in the file and at then end. As with the block format, carriage returns after the name (and first number) of the sequence signifies missing data.

Program reporting and troubleshooting:

In UNIX, POY prints output to stdout and progress information to stderr. In order to save the output, it must be redirected to a file (`> outputfile`). The stderr output can be silenced via the command `-noverbose` or redirected to a file as well (`2> errfile`).

When multiple data files are used in a simultaneous analysis, the taxon names in each file must match exactly (including capitalization). If the taxon names match you will see something similar to this:

```
input file aster.txt
input file data1.txt
input file data2.txt
input file data3.txt
change 1
collapse true
discrepancies true
graphics false
gap 2
intermediate false
maxtrees 1073741823
spr true
state true
tbr true
unpack_binary true
```

```
  verbose true (obviously)
This is poy, native code version 2.7-02-08-01-p2
Hennig86 file aster.txt: 91 characters, 10 taxa, weight 1
''
genbank file data1.txt: 10 taxa, weight 1
genbank file data2.txt: 10 taxa, weight 1
genbank file data3.txt: 10 taxa, weight 1
Non-Additives are done (19)
Made bin characters
Seed 1
Random replicate 0
```

If there are taxon mismatches or nonsense characters in a file you won't see random replicates starting. An error message is displayed and the program aborts. The order of taxa in the data files is not important except for specification of the outgroup (but see command -outgroup). As discussed above, the number of taxa must be the same and the names must be exactly the same and the program must find data and a return. If data is missing for some taxa in some data files use the taxon name plus two returns. The following testing script that doesn't run POY all the way through but rather tests the files rapidly is useful to check if all input files are in a correct format (one can modify this to input your files and examine the data*.err files to see if they are POY legal).

```
for FILE in data1.flat data2.flat data3.flat
  do
    poy -nobuild $FILE > $FILE.out 2> $FILE.err
  done
```

Once you have the data POY legal you will see program reporting indicating it is running. Picking up where we left off at the first replicate:

```
Random replicate 0
First optimize...outgroup is ASTERIAS, add node COSCINASTERIAS ->59
add node 3/10 CERTONARDOA discrepancy: real cost 132 calculated cost 150
->132 1 tree
add node 4/10 PSEUDARCHASTER ->182 1 tree
add node 5/10 ASTROPECTEN discrepancy: real cost 240 calculated cost 243
->240->239discrepancy: real cost 225 calculated cost 234
->225discrepancy: real cost 225 calculated cost 226
 2 trees      and so on…
```

Try the sample scripts, matricies and datafiles in ftp:/s/ftp.amnh.org/pub/molecular/poy

POY invoked by these simple scripts will do a series of random replicates followed by SPR and TBR swapping. The output will include treelength calculations as taxa are added during a build and treelength calculations during swapping. Also reported are discrepancies between heuristic calculations of treelength (a shortcut applied to all trees examined that provides a substantial speedup) and more accurate tree lengths (calculated for a subset of trees as a checkpoint). A per replicate shortest treelength is reported and a running total of how many short trees have been collected is given after each replicate. When all replicates are complete statistics are appended to the

error file. The output file includes all the shortest trees (which may include polytomies) and if `-spewbinary` is on the binary representations of the shortest trees (polytomies resolved and siblings ordered as to ensure that the same tree length will be found when these trees are used as inputtrees in POY) are included. You will need to use the binary trees also for any cross-checking of POY results against other software. Additional reporting occurs in parallel execution concerning the status of controllers, the progress of each replicate including refinement features such as tree fusion and drifting, and fault tolerance statistics.

Heuristic efficiency and execution time:

Discrepancies should be monitored for large differences between heuristic and checkpoint treelength calculations. If discrepancies are large, the user can address this by adjusting slop values for various tree building, and tree refinement procedures `(e.g., -buildslop n, -checkslop n,-ratchetslop n;` alternativley one can sets all slop values if they are not individually tuned `-slop n)`. Slop values can be used to examine suboptimal trees near the length of the putative optimal tree at the expense of greater execution times (see below for implementation details with each command).

Tree buffers:

Indecisive or missing data can lead to many equally parsimonious resolutions that occupy tree buffers and slow searches considerably due to swapping on many trees from a particular build or random replicate (searching in a single region of tree space). Often trees generated in a single replicate differ only in the placement of a few taxa. As a result the full diversity of topologies that would be generated in swapping on many trees per build can be found by swapping on a random subsample of these trees (-fitchtrees). CPU time is better spent for a more global search of tree space by the use of many replicates and aggressive refinement of the best tree(s) from replicates based on which hold only a few trees. Thus one should generate many replicates, swap on a few randomly sampled trees within each replicate, collect the best trees and submit them to further refinement.

A random subsample of trees generated in a build can be kept in POY using `-fitchtrees and a "maxtrees" command`. The command `-buildmaxtrees n` sets buffers to hold n trees within build replicates. The command `-holdmaxtrees n` sets the overall hold buffer from all replicates to allow the accumulation of trees from all builds. If `-holdmaxtrees n` or `-buildmaxtrees n (`or other tree buffers for other operations as outlined below) are not specified `-maxtrees n` sets the overall buffer limit.

As described above, POY invoked by the simple script `poy datafile > outfile` will do a random build replicate with the first taxon as the outgroup followed by TBR swapping. See command summary for tuning of options and parallel behaviors. A more aggressive and parallel search would include a mix of the commands below. Regardless of the order in which they are listed in the command line they are summarized here in the order in which POY will execute them within a replicate.

1) Perform initial build (stepwise addition of taxa), consulting options (e.g, `-buildslop, -buildmaxtrees n)`. If running in parallel, spawning jobs to slave nodes, consulting options `(-parallel, -multibuild n`).

2) Submit best topologies(s) held in buffers from building to SPR and TBR swapping consulting options (e.g., `-cutswap, -sprmaxtrees n, -intermediate`).

3) Submit best topologies held in buffers from swapping to tree drifting plus drifting options (e.g., `-driftspr, -numdriftspr n, -numdriftchanges n).`

4) Submit best topologies held in buffers from tree drifting to parsimony ratcheting plus ratcheting options (e.g., `-ratchetspr n, -ratchettbr n, -multiratchet`).

5) Submit best topologies held in buffers from ratcheting to tree fusion plus fusion options (e.g., `-fusemingroup n -fuselimit n`).

6) If `-random n` is set to n > 1 submit best toplogies held in buffers to a final round of TBR swapping consulting options (e.g., `-checkslop`).

7) Output best toplogies if `-repintermediate` is invoked.

8) Proceed to additional replicates.

9) Stop run and output shortest trees when `-minstop n` and `-stopat n` are satified or all replicates are complete.

Evaluating results.

Consensus trees can be calculated directly with the command `–printtree` or from POY output files, using the program JACK2HEN or putting the topologies in your tree viewer of choice.

JACK2HEN creates a consensus tree of n% as in 100 for strict consensus and lower numbers for more catholic structures.

```
jack2hen n < poy_output_file > tree_outputfile
```

Bremer values in POY can be calculated for all groups at once by TBR swapping under a set of constraints designated by a set of group inclusion characters (`-bremer -constrain`). A group inclusion matrix (one character for each member for a clade) is calculated via the program JACK2HEN.

Global Bremer support calculations may lead to overestimates of group support because POY calculates these values based on a single TBR branch swap sample. Informative trees more than a single TBR swap away will not contribute to the Bremer numbers. As an alternative to `-bremer` a series of `-disagree` searches each with a file constraining only one group at a time can be used to increase accuracy of Bremer numbers (employed in Frost et al., 2001). Bremer supports for each group are thus calculated individually in a series of runs. To accomplish this a group inclusion matrix is split into individual matrices of one character for each group. Each of these matrices is used as constraint files in individual POY runs for each group to search for the shortest tree that did not include the group (`-disagree –constrain filename`). Bremer values can be calculated as the difference between the number of steps in the constraint tree and the best tree for a dataset. See `-disagree` in the command summary below for examples.

Commands that specify strategy and buffers for tree building and random replication:

```
-approxbuild
-approxquickspr
-approxquicktbr
-build
-buildmaxtrees
-buildslop
-buildspr
-buildtbr
-jackboot
-jackstart
-multibuild
-oneasis
-random
-randomizeoutgroup
-seed
-topofile
-topology
```

Commands to tune the accuracy of the heuristics of tree length calculations for overall search (see other "slop" type options mentioned in refinement and building categories):

```
-checkslop
-exact
-slop
-quick
```

Commands that set tree buffer for overall search (see other "maxtree" type options mentioned in refinement and building categories):

```
-fitchtrees
-holdmaxtrees
-maxtrees
```

Commands that modify the treatment of data or constraint files:

```
-agree
-bremer
-change
-compressstates
-constrain
-defaultweight
-disagree
-dp
-dropconstraints
-extensiongap
-fixedstates
-gap
```

```
-goloboff
-goloboff ck
-goloboff cm
-goloboff ri
-kfactor
-leading
-molecularmatrix
-multiplier
-noconstrain
-prealigned filename
-recode
-trailinggap
-topofile
-topology
-weight
```

Commands that modify the treatment of data or constraint files under likelihood:

```
-basefreq
-estimatep
-estimateparamsfirst
-estimateq
-freqmodel
-gammaclasses
-gammalpha
-invariantsitesadjust
-likelihood
-likelihoodconvergancevalue
-likelihoodestimationsize
-likelihoodesttranseachtime
-likelihoodextensiongap
-likelihoodmaxnumiterations
-likelihoodroundingmultiplier
-likelihoodstep
-likelihoodtrailinggap
-printqmat
-qmatrix
-submodel
-theta
-trullytotallikelihood
```

Commands that modify the optimization procedures (e.g., fixed states, search based, and iterativepass):

```
-fixedstates
-compressstates
-hypancfile
-hypancname
```

```
-iterativeinitfinal
-iterativeinitsingle
-iterativekeepbetter
-iterativelowmem
-iterativepass
-iterativepassfinal
-iterativerandom
-maxiterations
-newstates
-pairmatrix
-printhypanc
-printlotshypanc
-showiterative
```

Commands that control ratcheting:

```
-checkfrequency
-ratchetoverpercent
-ratchetpercent
-ratchetseverity
-ratchetslop
-ratchetspr
-ratchettbr
-ratchettrees
```

Commands that control tree drifting:

```
-approxdrift
-driftequallaccept
-driftlengthbase
-driftmetric
-driftrees
-driftspr
-drifttbr
-numdriftchanges
-numdriftspr
-numdrifttbr
```

Commands that control tree fusion:

```
-fuseafterreplicates
-fuselimit
-fusemaxtrees
-fusemingroup
-fusingrounds
-treefuse
-treefusespr
-treefusetbr
```

Commands that control branch swapping:

```
-approxquickspr
-approxquicktbr
-cutswap
-spr
-sprmaxtrees
-tbr
-tbrmaxtrees
```

Commands that control dynamic process migration in computing clusters:

```
-dpm
-dpmacceptratio
-dpmautoadjustperiod
-dpmjobspernode
-dpmmaxperiod
-dpmmaxprocessors
-dpmminperiod
-dpmperiod
-dpmproblemsize
```

Commands that modify parallel behavior of options and load distribution in a cluster or SMP machine:

```
-controllers
-jobspernode
-maxprocessors
-minstop
-multibuild
-multidrift
-multirandom
-multiratchet
-onan
-onannum
-parallel
-randomizeslaves
-solospawn
-stopat
```

Commands that report on given trees:

```
-bremer
-build
-diagnose
-topology
```

Commands that specify progress, result reporting, and output options:

```
-characterweights
-discrepancies
-impliedalignment
-impliedalignment
-indices
-intermediate
-repintermediate
-stats
-time
-verbose
```

Command summary (alphabetical):

Each of these commands is preceded by a dash "**-**" when entered on the command line. Most binary commands can be disabled by preceding the command name with "no" (e.g. `-nospr`, which turns off `-spr`).

`-agree` *`filename`* `[agree -disagree]`: Find shortest tree that agrees with the groups in the constraint file when performing a search. The constraints file can be made manually, or via the program JACK2HEN (ftp.amnh.org/pub/molecular/poy) to convert parenthesis notation to a HENNIG86 style constraint file.

`-approxbuild` `[noapproxbuild -noapproxbuild]`: Performs initial tree building using an approximate shortcut because it skips the correction for heuristic tree calculation errors during the build process. If the user finds that build results are unsatisfactory the option `-noapproxbuild` will take longer but do a better quality initial build. This command is highly recommended, especially if `–multibuild` and/or `–multirandom` are used.

`-approxdrift` `[approxdrift -noapproxdrift]`: Does not correct for heuristic tree calculation errors during the treedrifting process. If the user finds that drift results are unsatisfactory the option `-noapproxdrift` will take longer but do a better quality initial drift.

`-approxquickspr` `[approxquickspr –noapproxquickspr]`: Performs quick SPR during initial tree building using an approximate shortcut.

`-approxquicktbr` `[approxquicktbr -noapproxquicktbr ]`: Performs quick TBR during initial tree building using an approximate shortcut.

`-basefreq` *`filename`* `[if unspecified poy will estimate frequencies as specified by -estimatep]`: Read base and indel frequencies from a file for use in likelihood analysis. See `-likelihood` . The order is A C G T INDEL and the file must end with a semicolon. For example:

```
0.2  0.1  0.3  0.35  0.05
;
```

-bremer [nobremer -nobremer]: Prints Bremer support values based on a TBR search—not via collapsing ever more catholic searches—hence these values may overestimate group support. The **-bremer** command requires a constraint file. Each of the characters in the constraint file signifying a group. This can be created from an output topology via JACK2HEN. If POY reports negative Bremer values that means shorter trees have been found. Increase**-checkslop** values and rerun original searches. Example command line:

```
poy -bremer -constraint constraintfile datafile > outfile 2> errfile
```

-build [build –nobuild]: Builds cladogram, mainly used as **-topology -nobuild** so that only diagnosis of a topology occurs without proceeding to cladogram construction.

-buildmaxtrees *n* [by default this command uses argument for **-maxtrees**]: Set maximum number of trees held in buffers during the building process.

-buildslop *n* [0]: Sets the slop value to n for the build process only. If not specified, the overall slop value, as specified by **-slop** n is used during the build process.

-buildspr [nobuildspr -nobuildspr]: Appends single tree SPR search to initial build process.

-buildtbr [nobuildtbr -nobuildtbr]: Appends single tree TBR search to initial build process.

-change *n* [1]: Set nucleic acid base substitution cost.

-characterweights [nocharacterweights –nocharacterweights]: Prints individual character information such as length, minimum length, maximum length, weight, and fit statistics.

-checkfrequency *n* [0]: Sets frequency of message checking in parallel ratcheting (multiratcheting only).

-checkslop *n* [0]: By adding an extra round of TBR branch swapping, this command checks all cladogram lengths that are within *n* tenths of a percent of the current minimum value. For example, a slop value of 10 would check all cladograms found within 1% of the minimum tree length. This command slows down the search, but is less effected by the heuristics of the tree length calculation shortcuts. Use in conduction with low slop values such that shorter trees may be found more efficiently by concentrating on the TBR stage of the search e.g., **-slop 5 -checkslop 10**.

-clist *xyz* [null]**:** Print a list of all commands that have usage information that cross-references any command that contains the string **xyz**.

See also **-help, -list, -list, -cllist, -dlist, -dllist**

-cllist *xyz* [null]**:** Print a list of all commands that have usage information that cross-references any command that contains the string xyz, and include the usage information of the commands in this list.

See also -help, -list, -llist, -clist, -dlist, -dllist

-commandfile *filename*[poy.ini]**:** In the command line, substitute the content of file filename for itself. (the default is to look in the users current directory for file poy.ini).

For example with file PLOTSTRICT containing:

```
-printtree -plottrees off -plotmajority off -plotstrict on –plotfile
stdout
```

the command line

```
poy mydata -commandfile PLOTSTRICT -random 5
```

is equivalent to

```
poy mydata -printree -plottrees off -plotmajority off -plotstrict on -
plotfile stdout -random 5
```

A single run can have multiple -commandfile commands, and the commandfiles themselves may have embedded -commandfile commands. Nesting is unlimited as long as there is no circular recursion.

As with other input files, commandfiles must be text files that are properly formatted for the OS that you use (but see -enabletmpfiles). There are no further restrictions on format (there can be multiple lines with multiple commands each).

"--" (double dash) can be used as shorthand for -commandfile

For example:

```
poy mydata -- PLOTSTRICT -random 5
```

```
poy mydata -commandfile PLOTSTRICT -random 5
```

are equivalent (note the space between "--" and "PLOTSTRICT").

POY always checks if there is a file in the current directory that is called "poy.ini". If that file exists, it is treated as a commandfile, and its contents come at the beginning of the command line. There is no possibility to turn poy.ini off.

For example with file poy.ini containing

```
-parallel
```

the command line

```
poy mydata -- PLOTSTRICT -random 5
```

will expand to

```
poy mydata -parallel -printree -plottrees off -plotmajority off -
plotstrict on -plotfile stdout -random 5
```

In order to keep track of commands that are actually in use, both the original and the fully expanded commandline are output to stderr at the start of a run.

See also `-commandfiledir, --`

`-commandfiledir` *dirname* `[current directory]:` Set the directory where POY looks for commandfilenames. There can be multiple `-commandfiledir` commands, each one determining the directory for commandfiles in the remainder of the commandline as it is scanned from left to right.

The default commandfile `poy.ini` is expanded at the start of the commandline and is therefore not addressed by `-commandfiledir`. So `poy.ini` must always be in the current directory.

For example if the directory /home/me/myshorthands contains a file named `PWC` with the contents:

```
-phastwincladfile
```

and a file IA that contains

```
-impliealignment
```

and the current directory contains a file `IA_TO_FILE` with contents

```
- commandfiledir /home/me/myshorthands/ -- IA -- PWC
```

then

```
poy mydata -- IA_TO_FILE myimpliedalignment
```

will expand to

```
poy mydata -impliedalignment -phastwincladfile > myimpliedalignment
```

which, on the basis of the best trees that are found using the default tree search strategy, will calculate an implied alignment and output it to file `myimpliedalignment`.
When dealing with a `commandfiledirectory` and a `commandfile`, POY does not check if the directory is a valid directory or if the file is a valid file in that directory. POY only checks if the literal concatenation of the two is a valid file. This is why "/home/me/myshorthands/" needs the "/" at the end. Without the trailing / POY would notify you that file

"/home/me/myshorthandsmycommand" does not exist and then exit. A related command is -datadir, which is used to set the directory of inputfiles other than commandfiles.

See also -commandfile, --, -datadir

-compressstates [nocompressstates -nocompressstates]: During search-based-optimization, notes the states used during the build stage and removes unused states from the state set (as determined by -newstates *filename*) during swapping. The state set is returned to its original size for a final round of swapping if -checkslop n is specified. This can speed up the search with the tradeoff of potentially suboptimal solutions.

For example

```
poy -fixedstates datafile -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt -newstates datafile.states -compressstates > outfile 2>
errfile
```

See also -printhypanc

-constrain *filename* [constrain –noconstrain]: Constrain the search to conform to the group inclusion characters specified in the file *filename*. **This file must be a HENNIG86 file with binary characters**. The default is to agree with the constraints. All input files after the -constrain command are viewed as constraint files until the counter command -noconstrain is invoked (or more simply just put the constraint file at the end of the command line). One way to estimate Bremer support values for clades, -constrain filename can be employed with -bremer (but see also -disagree).

An example of group inclusion characters for each clade:

```
xread
''
2 4
frog     11
chicken  11
cicada   10
mushroom 00
;
cc- 0.1;
proc /;
```

-controllers *n* [1]: Number of subclusters in parallel search. With ns the number of slave nodes, these ns nodes are divided into n groups of ns / n; the remaining ns mod n slaves are assigned to the first ns mod n subclusters. In concert with **-multirandom –random n** simultaneous randomized full searches are executed in parallel within each subcluster. For most efficient control groups can be found by benchmarking your cluster. For a cluster of 256 Intel PIII 500s over 100 Mbit Ethernet, refinement procedures were efficient for ~32 processors and these values are used as follows (Janies and Wheeler, 2001). If a user has 256 processors at his disposal then 256 / 32 = 8 and **-controllers 8** is set providing efficient hierarchical parallelism of 8 subclusters of 32 processors each.

Typical command lines in this configuration:

```
poy -parallel -controllers 8 -multirandom -random 8 -multibuild 32
datafile > outfile 2> errfile
```

This command line performs eight full searches (build through refinement) in parallel. Each search would have 32 builds (for a total of 256 builds); the best of which are submitted to further refinement. This is likely the most efficient way to use a Beowulf class cluster like the one described above.

Alternatively:

```
poy -parallel -controllers 8 -multirandom -random 64 datafile > outfile 2>
errfile
```

This command line performs 64 full searches (build plus refinement). One full search will be spawned in each of the 8 subclusters. As each subcluster completes a full search that subcluster will receive another search replicate until all 64 replicates are complete. See also **-minstop -multirandom** and **-stopat.**

-cutswap [cutswap –nocutswap]: Shortcut for branch swapping (not used in building). The negation **-nocutswap** makes for a more exhaustive and time consuming search. When the discrepancies are large this command may cut off profitable search paths.

-datadir *dirname* [current drectory]**:** Set the directory where POY looks for inputfiles other than commandfiles. These can be files with actual data or files with other information (e.g**. -molecularmatrix**). To set the directory for commandfiles, use **-commandfiledir**.

For example, running POY from /home/me/workdir and with datafiles its1a, its1b,its2a, its2b, and its2c in directory /home/me/gentians/Lisianthius/its

```
poy -datadir /home/me/gentians/Lisianthius its1a its1b its2a its2b its2c -
random 5 -datadir ./ -molecularmatrix g1tv1ts1.txt
```

expands to

```
poy -datdir /home/me/gentins/Lisianthius
/home/me/gentians/Lisianthius/its1a /home/me/gentians/Lsianthius/its1b
/home/me/gentians/Lisianthius/its2a /home/me/gentians/Lisianthius/its2b
/home/me/gentians/Lisianthius/its2c -random 5 -datadir / -molecularmatrix
./ g1tv1ts1.txt
```

This reduces typing effort and allows one to organize datafiles outside of the working directory. To keep track of everything, the original and fully expanded commandline is output to stderr at the start of a run.

See also `-commandfiledir, -molecularmatrix`

`-defaultweight` *n* [1]: Sets the weight of all succeeding input files to n. This can be used with an argument of -1 to search for the worst possible tree for a dataset. This maximum tree length is necessary for rescaled ILD (Wheeler and Hayashi, 1998) and meta RI calculations (Wheeler et al., in prep).

`-deletegapsfrominput` *n* [20]: For sequence files in FASTA format, use only the first stretch of non-blanks after > as taxonname.

This allows to enter some annotation on the > line

example: with `-fastashortname`

```
> taxona some comments on this sequence
aaacgt
aac
> taxonb some comments on this sequence
aaacgt
```

will be read as a sequence `aaacgtaac` for taxona and a sequence `aaacgt` for taxonb

example: with `-nofastashortname`

```
> taxona some comments on this sequence
aaacgt
aac
> taxonb some coments on this sequence
aaacgt
```

will be read as a sequence `aaacgtaac` for taxona_some_coments_on_this_sequence and a sequence `aaacgt` for taxonb_some_comments_on_this_sequence

The general format of a FASTA sequence file is:

A line with a taxonname starts with > (the > has to be first character on the line) and every following line (blank lines included) until the next line with a name or until the end of the file is taken as sequence data for the preceding name. The sequences must consist of valid IUPAC codes but in poy there are some optional characters. Sequences can now have the characters # and * in them. These characters are used to indicate multiple fragments in single files. # is the fragment separator, * can be used to exclude individual fragments from the analysis.

Compare this to the other format for sequences that poy uses (some constraints are removed now compared to older poys): Entries for taxa are separated by an empty line (i.e. a line that consist of blanks only - so spaces are allowed). In the new format, the first stretch of non-blanks in a file is considered the first taxonname, and all following lines until end-of-file or until an empty line occurs, are read as data for terminal. The first stretch of non-blanks on the first following non-blank line is the name for the second taxonname, and so on. In this case, the data can have interspersed integers (cf old BLOCK format), but these integers ars NOT read as part of the data.

For example:

```
>a
acc#aacgt-t#tttttt#a*t
>b
*#aacgtcc#ttt*tt#
```

Is read as involving four fragments for taxa a and b.

```
          frag1     frag2    frag3    frag4
taxon a   acc       aacgt_t  tttttt   at
taxon b   missing   aacgtcc  ttttt    missing
```

Of these, only fragment 2 is used (the other are excluded with the *); however with `-deletgapsfrominput`, the second fragment for taxon a would be read as `aacgtt`. A possible use is the following heuristic search strategy. Output some implied alignments for some decent set of trees, minimally edit these alignments to indicate fragments (possibly eliminating fragments, such as non-homologous stretches at the beginning or end of sequences). Use the result as input for a next round of (constrained) search (the constraint here is in the fragment delimitation).

Note that # is a fragment separator (as opposed to delimiter):

```
>a
#aac#aat#
>b
#aat#ac#
```

Involves four fragments: the two that are explicitly entered, a third one before the first #; and a fourth one following the third #. These last two are considered missing in both and b.

Each taxon must have the same number of # (1 less than the number of fragments) otherwise POY stops with an error message.

The * can appear anywhere in a fragment, in any taxon for that fragment.

see also **-printccode** for formatting requirements for morphological data, **-fastashortname**, **-terminalsfile, -minterminals.**

-diagnose [nodiagnose –nodiagnose]: Prints branch lengths and apomorphy lists for cladograms derived form a search or input with –topology (employed in Frost et al., 2001). If **-diagnose** is not specified, only the cladogram length is reported.

-disagree [agree -agree]: Use with **-constrain filename** to contradict with constraints to provide shortest trees that are not included in constraint file. One application is for Bremer support tests (employed in Frost et al., 2001) on a group by group basis. This is an alternative to the **-bremer** command that produces values for all groups simultaneously. For example, for Bremers for each group in the example constraint file given above for **-constrain** you must cut the characters that represent single groups and create two new files each used for an individual search:

```
poy data -random 10 -norandomizeoutgroup -disagree -constrain
constraintfile1 -molecularmatrix matrixfile > outfile1 2> errfile1
```

constraintfile1 contains:

```
xread
'first group'
1 4
frog     1
chicken  1
cicada   1
mushroom 0
;
cc- 0.0;
proc /;
```

```
poy data -random 10 -norandomizeoutgroup -disagree -constrain
constraintfile2 -molecularmatrix matrixfile > outfile2 2> errfile2
```

constraintfile2 contains:

```
xread
'second group'
1 4
frog     1
chicken  1
cicada   0
mushroom 0
;
cc- 0.0;
proc /;
```

-discrepancies [discrepancies –nodiscrepancies]: Print difference between cladogram length from shortcuts and complete down pass.

-dlist *xyz*[null]**:** Print a list of all commands that have usage information that contains the string xyz.

See also -help, -list, -llist, -clist, -clist, -dllist

-dllist *xyz*[null]**:** Print a list of all commands that have usage information that contains the string xyz, and include usage information.

See also -help, -list, -llist, -clist, -cllist, -dlist

-dp [nodp –nodp]: Perform complete Sankoff-style dynamic programming based on molecular matrix, additive or nonadditive matrices on data entered in HENNIG86 format.

-dpm [nodpm –nodpm]: In a parallel environment, use dynamic process migration to move processes form relatively over utilized processors to relatively less active processors. This is used to improve load balancing in cluster computers. It may improve speed of calculations significantly when more than one POY script is run simultaneously or when the cluster consists of processors with significantly different speeds. See other –dpmxxx commands for more information.

-dpmacceptratio *n* [1.5]: Threshold in the ratio of instantaneous processor performance to trigger a task migration under dynamic process migration (see –dpm). The performance is a combined measure of processor load and intrinsic processor speed. When the master node or any of the controllers need a slave task to perform some calculations, POY calculates the performance of the first available regular job and the performance of all current reserve jobs. If the ratio of performances of the best reserve job and that regular job is at least *n*, the regular job is relegated to the pool of reserve tasks while the best reserve job is recruited into the pool of regular jobs. The work to be done is then sent to the new regular job. In practice, these calculations do not occur on every possible occasion; but with intervals of –dpmperiod (the 'bid' period).

-dpmautoadjustperiod [nodpmautoadjustperiod -nodpmautoadjustperiod]: Under dynamic process migration (see -dpm), automatically adjust the bid period to send out requests to the reserve jobs for calculations of performance. If these requests lead to task migration, the period is reduced. If bids are unsuccessful, it is increased.

-dpmjobspernode *n* [1]: Under dynamic process migration (see -dpm) the number of reserve jobs spawned to allow task migration. Operates akin to –jobspernode.

-dpmmaxperiod *n* [100]: Under dynamic process migration (see -dpm), the maximum bid period allowed under -dpmautoadjustperiod.

-dpmmaxprocessors *n* [number or available machines in cluster]: Under dynamic process migration (see -dpm), the number of reserve jobs to spawn. Operates akin to –maxprocessors.

-dpmminperiod *n* [1]: Under dynamic process migration (see -dpm) the minimum bid period allowed under -dpmautoadjustperiod.

-dpmperiod *n* [10]: Under dynamic process migration (see -dpm) the bid period (or starting bid period under –dpmautoadjustperiod).

-dpmproblemsize *n* [5000]: Under dynamic process migration (see -dpm) the size of the calculation that is used to assess the performance of a job on a particular processor. The bigger the n, the larger the calculation and the better the assessment of performance, more time used to generate this information.

-driftequallaccept *n* [50]: Percentage of time to accept equally parsimonious trees during treedrifting. See also –drifttbr –driftspr.

-driftlengthbase *n* [2]: 1/(n+c-b), let c=length of candidate tree, b=best tree thus far. In treedrifting this represents the probability of accepting a suboptimal tree.

-driftmetric *s* [length]: let s = rfd, length2, or length. This is the metric for treedrift probablities. See also –drifttbr –driftspr.

-driftspr [nodriftspr -nodriftspr]: Performs one round of treedrifting based on SPR search (Goloboff, 1999).

-drifttbr [nodrifttbr -nodrifttbr]: Performs one round of treedrifting based on TBR search (Goloboff, 1999).

-dropconstraints [nodropconstraints –nodropconstraints]: Abandon constraints after build and before swapping. This is used to swap on an input tree, derived from constraints, but do an unconstrained swap.

-enabletmpfiles [noenabletmpfiles -noenabletmpfiles]: This command currently only works under Linux (not under Windows and not currently under MacOS X). With temporary files enabled, POY will create and use two temporary files in the current directory. The names of these are hardcoded and are poy1.tmp and poy2.tmp (so running two poy jobs with –enabletmpfiles simultaneously in the same directory is a good recipe to have them both crash). If such files exist, they are overwritten without warning. The temporary files are used to do some input preprocessing. Currently this consists of the following:

1. convert MacOS X and dos file formats to Unix file formats
2. make the format requirements for trees in parenthetical notation (-topology and -topofile) less stringent: single trees may be on multiple lines; there must be no space between the end of a taxon name and a closing parenthesis (as a consequence, taxonnames are not allowed to contain parentheses); trees may be delimited by a * instead of [n] (the Hennig86 and Nona tread convention)
3. issue sensible error messages when an error is encountered while reading a tree in parenthetical noation

In addition, -enabletmpfiles changes the behavior of the –topology command, which can now be followed by multiple trees instead of just one tree. The different trees must separated by [n] (n stands for an integer) or by * and the last tree must be followed by a semicolon (optionally preceded by [n]).

-estimatep [estimatep –noestimatep]: Under –likelihood, estimate base and indel frequencies from input data and pairwise alignments and fix for the duration of the search. When -noestimatep is invoked, base frequencies are estimated for each pair of sequences as they are encountered.

-estimateparamsfirst [estimateparamsfirst –noestimateparamsfirst]: Under -likelihood, estimate parameters from input data and pairwise alignments and fix for the duration of the search. When -noestimateparamsfirst is invoked, parameters are re-estimated for each pairwise sequence comparison.

-estimateq [noestimateq –noestimatep]: Under –likelihood, estimate transition probabilities from pairwise alignments and fixed for the duration of the search.
When -noestimatep is invoked, transition probabilities are estimated for each pair of sequences as they are encountered.

For example, Basic ml search (estimate base and indel frequencies):

```
poy datafile -seed -1 -random 10 –likelihood > outfile 2> errfile
```

alternatively, estimate and fix the q matrix (transition probabilities) before a search.

```
poy –estimateq datafile -seed -1 -random 10 –likelihood > outfile 2>
errfile
```

alternatively, estimate p and q for each branch of each candidate tree during a search.

```
poy –noestimatep datafile -seed -1 -random 10 –likelihood > outfile 2>
errfile
```

-exact [noexact –noexact]: Exact treelength calculation based on the heuristic homologies of the downpass (or iterativepass). Used to improve treelength calculations whenever slop features find a better or equal length tree during a search. Sacrifices execution speed for accuracy.

-extensiongap *n* [gap cost]: Affine or extension gap cost. Normally, the cost of a gap of length m would be m*gap cost as assigned by –gap. With –extensiongap the initial gap costs –gap (or that set by –molecularmatrix) and any other contiguous gaps cost -extensiongap. Hence, length m gap would cost, (gap cost + (m-1) * extension gap cost). For example:

```
poy datafile -seed -1 -random 10 –molecularmatrix g2tv1ts1.txt
–extensiongap 1 > outfile 2> errfile

poy datafile -seed -1 -random 10 –gap 2 –extensiongap 1 > outfile 2>
errfile
```

-fastashortname [fastashortname, -nofastashortname]: For sequence files in FASTA format, use only the first stretch of non-blanks after > as taxonname. This allows to enter some annotation on the > line. See also **-deletegapsfrominput for a full discussion of use of FASTA formats.**

-fitchtrees [nofitchtrees –nofitchtrees]: Ensures that trees kept in buffer (set by –maxtrees) are a random subset of those that would have been kept if the treebuffer would have been larger. This is based on the algorithm of W. Fitch (pers. comm).

-fixedstates [nofixedstates –nofixedstates]: Optimizes sequence data using the fixed states procedure of Wheeler (1999). Must be invoked before data file to be considered under fixed states; and all names of datafiles that follow are optimized using fixed states until the command is turned off using –nofixedstates. Also used in search based optimization; See also –newstates.

```
poy –fixedstates datafile -seed -1 -random 10 > outfile 2> errfile
```

-freqmodel s [f5 is the default, user can choose f5 f2 or f1]: Under -likelihood, number of base frequencies to estimate. f5 allows all base (and indel) frequencies to vary, f2 allows two classes (bases and indels) , and f1 sets all to 0.2.

-fuseafterreplicates [nofuseafterreplicates -nofuseafterreplicates]: Performs tree fusing on the results of random replicates as they come in.

-fuselimit *n* [default is to hold all pairwise trees that memory will allow]: Limits the number of tree fusing pairs to n. By default all possible pairwise fusings are done (this is equal to t^2-t ; let t = number of trees).

-fusemaxtrees *n* [default is to hold all trees that memory will allow]: Limits the number of tree fusing trees saved to n.

-fusemingroup *n* [5]: Minimum number of taxa in a subtree to be exchanged in tree fusion.

-fusingrounds *n* [2]: Performs tree fusing and fuses the resulting trees n times.

-gammaalpha *n* [not set by default]: Under -likelihood if specified, the value of the alpha parameter of the gamma distribution, if the user chooses to set gamma classes > 0.
For example:

```
poy datafile -seed -1 -random 10 –likelihood –gammaclasses 2 –gammaalpha
0.1 > outfile 2> errfile
```

-gammaclasses *n* [0]: Under -likelihood the number of rate classes modeled under the discrete gamma.

-gap *n* [2]: Set insertion-deletion cost (a.k.a. gap cost).

-goloboff *n* [null]: sets implied character weighting ala Goloboff (1993). The mode of weight is specified as "ck," "cm," or "ri." The command **-goloboff ck** will perform implied weighting as described in Goloboff's original paper ((k + 1)/(k + 1 + s + m)). The command **-goloboff cm** modified Goloboff's weight by a factor of the minimum character length ((k + 1)/((k + 1 + s + m)/m)). This has little effect on morphological characters, but makes more sense with molecular characters which may have hundreds of steps separating minimum and maximum character length values. The command **-goloboff ri** weights via the retention index. Since these weights decrease cladogram length by reducing character weights, a factor (set by "multiplier") is used. Character lengths are multiplied by "multiplier" to allow subtle distinctions in weighted cladogram length (and keep the calculations integer). As a result, lengths will appear longer by the "multiplier" factor. The default for "multiplier" is 100.

-help [nohelp nohelp]**:** Print an overview of commands for getting more information. Same as **--help.**

-holdmaxtrees *n* [hold all trees available that memory will allow]: Number of trees to hold over all random replicates.

-hypancfile *filename* [poy.hypanc]: Used with –printhypanc and –newstates to assign a user specified filename for hypothetical ancestral state output. Distinct from the output of –diagnose in that ambiguously optimized ancestral states are resolved. This is used to generate state sets for search-based-optimization. See also **-newstates,** for example**:**

```
poy -fixedstates datafile –newstates datafile.states -seed -1 -random 10 -
molecularmatrix g1tv1ts1.txt –printhypanc –hypancfile otherfilename  >
outfile 2> errfile
```

-hypancname [nohypancname **-nohypancname**]: Print names of HTU in hypancfile output used with –newstates, for example:.

```
poy -fixedstates datafile –newstates datafile.states -seed -1 -random 10 -
molecularmatrix g1tv1ts1.txt –printhypanc -hypancname > outfile 2> errfile
```

-impliedalignment [noimpliedalignment –noimpliedalignment]: Generates a topology specific multiple alignment based on the synapormorphy scheme (i.e. apomorphy list that can be output using **-diagnose**).

-indices [noindices –noindicies]: Prints out tree fit statistics (estimate of CI, RI).

-intermediate [nointermediate –nointermediate]**:** Prints intermediate search results. This is especially useful when performing long searches for which one may want to record job progress to disk. See also –repintermediate.

-invariantsitesadjust [noinvariantsitesadjust -noinvariantsitesadjust]: Under –likelihood, adjust likelihoods for theta fraction invariant sites.

```
poy datafile -seed -1 -random 10 –likelihood –invariantsitesadjust >
outfile 2> errfile
```

-iterativeinitsingle [noiterativeinitsingle -noiterativeinisingle]: Under -iterativepass, initialize HTU nodes with reconstructed final states with any ambiguities resolved.

```
poy -fixedstates datafile -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt –iterativepass -iterativeinitsingle > outfile 2> errfile
```

-iterativekeepbetter [iterativekeepbetter -noiterativekeepbetter]: Under -iterativepass, keep better HTU reconstructions between normal up/down pass and iterativepass.

```
poy -fixedstates datafile -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt –iterativepass -iterativekeepbetter > outfile 2> errfile
```

-iterativelowmem [noiterativelowmem -noiterativelowmem]: Under –iterativepass minimally allocate memory. This can be a big (factor of 100) savings in memory, but there is overhead. (Beta-There are known bugs in this option.)

```
poy -fixedstates datafile -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt –iterativepass -iterativelowmem > outfile 2> errfile
```

-iterativepass [noiterativepass -noiterativepass]: Perform 3-dimensional (using 3 vertices attached to any internal node) optimization-alignment to improve HTU estimation and tree length. Sequences must be similar in length.

-iterativepassfinal [noiterativepassfinal -noiterativepassfinal]: Under -iterativepass, use iterativepass to re-estimate final states.

-iterativerandom [noiterativerandom -noiterativerandom]: Under "iterativepass," reconstruct HTU final states but resolve ambiguities with random preference.

-jackboot [nojackboot –nojackboot]: perform Farris' parsimony jackknifing procedure (Farris, et al., 1996) with -random n pseudoreplicates (note that this changes the meaning of the –random command). The tree search strategy within each pseudoreplicate can be set by using the commands that affect trees search within regular –random replicates.

-jackfrequencies

For example:

```
poy datafile -seed -1 –random 200 –oneasis –multibuild 5 –tbr –buildspr >
outfile 2> errfile
```

instructs POY to construct and analyze 200 pseudoreplicates. For each pseudoreplicates, POY does five addition sequences (for the first, the taxon order is as in the first dataset; the following four use a pseudo-random order according to the –seed option), each followed by spr; the resulting best trees are subjected to tbr. The default output consists of the n strict consensus trees for the pseudoreplicates and a majority consensus trees of these strict consensus trees. The frequencies of

clades on this tree are the jackknife percentages. The default output can be changed with the `options –jackoutgroup, -jackpseudoconsenstrees, jackpseudotrees, jacktree, and jackwincladfile`. Tree search strategies that are too shallow often lead to severe underestimation of the jackknife support values.

`-jackoutgroup` *`taxonname.`* Specifies the single taxon to be used as outgroup for jackknifing. Defaults to the first taxon of the first dataset or to `–outgroup` if that command is specified.

`-jackpseudoconsensustrees` [`jackpseudoconsensustrees -nojackpseudoconsensustrees`] If on, and if `–jackboot and -nojackstart`, outputs the strict consensus trees for the individual pseudoreplicates (parenthetical notation; for each consensus tree the number of best trees on which it is based is indicated between square brackets)

`-jackpseudotrees` [`nojackpseudotrees -nojackpseudotrees`] If on, and if `–jackboot` and `-nojackstart`, outputs the individual best trees for all pseudoreplicates (parenthetical notation with length indication between square brackets).

`-jackstart` [`nojackstart -nojackstart`]: Use jackknifing to generate starting trees then proceed with normal search (`-jackboot` must be on as well).

`-jacktree` [`jacktree -nojacktree`] If on and if `–jackboot and –nojackstart`, output the majority rule consensus tree of the strict consensus trees of all jackknife pseudoreplicates (text-based graphics). The clade frequencies in this tree are the jackknife support values.

`-jackwincladfile` *`filename`* [none] If `–jackboot` and `–nojackstart`, create the file filename with a tread statement that describes the strict consensus trees of the individual pseudoreplicates in parenthetical notation.

`-jobspernode` *`n`* [`1`]: Parallel option. Spawns n POY processes per node. Often this is set to the number of processors per node (e.g. 1, 2, or 4).

`-kfactor` *`n`* [`0`]: sets the "k" value of Goloboff (1993) for implied character weighting.

`-leading` [`leading -noleading`]: Count leading and trailing gaps. Often leading and trailing gaps result from sequencing with various primers by various investigators and are not evolved differences in sequence length. In this case use `-noleading`. In use of `-noleading`, leading and trailing gaps are initially accounted for during builds to prevent a trivial nonoverlapping alignment but not counted when determining tree length (see also `-trailinggap`).

`-likelihood` [`nolikelihood -nolikelihood`]: Use likelihood as optimality criterion. Morphological characters are treated as in Tuffley and Steele (1997). Likelihoods are reported in –ln numbers.

`-likelihoodconvergancevalue` *`n`* [`1`]: Under `-likelihood` similarity threshold for two likelihood scores to be considered equal. That is, the difference in two likelihood must be less than this number to be considered equal. Increase in this value will speed up likelihood calculations. The

units are 1/likelihoodroundingmultiplier, which is set to 100 by default. Hence the default identity threshold would be 0.01 log likelihood units.

```
poy datafile -seed -1 -random 10 –likelihood –likelihoodconvergancevalue
2 > outfile 2> errfile
```

-likelihoodestimationsize *n* [default in parallel = all pairwise; in sequential = number of taxa]: Under -likelihood the number of pairwise sequence comparisons used to estimate likelihood parameters. In parallel, the default is all pairwise comparisons; in sequential mode the default is the number of taxa.

-likelihoodesttranseachtime [likelihoodesttranseachtime -nolikelihoodesttranseachtime]: Under -likelihood, estimate transition matrix (q) for each comparison between two sequences.

```
poy datafile -seed -1 -random 10 –likelihood –nolikelihoodesttranseachtime
> outfile 2> errfile
```

-likelihoodextensiongap *n* [not set]: Under -likelihood, increase in likelihood of extension (affine) gaps, Akin to –extensiongap n in parsimony analysis. This parameter is a means of including non-linear gap costs in likelihood analyses, but beware that it is a kludge rather than a true model parameter !

```
poy datafile -seed -1 -random 10 –likelhood –likelihoodextensiongap 0.5 >
outfile 2> errfile
```

-likelihoodmaxnumiterations *n* [100]: Under –likelihood , the maximum number of branch length iterations to be performed.

-likelihoodroundingmultiplier *n* [100]: Under -likelihood, internal optimization alignments are performed using double precision floating point arithmetic, but are converted to integers on return. This value sets the rounding value to 1/n. The default precision then 0.01 log likelihood units.

For greater precision one might specify:

```
poy datafile -seed -1 -random 10 –likelhood -likelihoodroundingmultiplier
10000 > outfile 2> errfile
```

-likelihoodstep *n* [5]: Under -likelihood, the size of iteration steps during branch length optimization. A larger number denotes a smaller step size.

-likelihoodtrailinggap *n* [not set]: Under -likelihood, increase in likelihood of leading and trailing gaps. Akin to -trailinggap in parsimony analysis, this parameter is a means of including differential gap costs in likelihood analyses, but beware that it is a kludge rather than a true model parameter !

```
poy datafile -seed -1 -random 10 –likelhood – likelihoodtrailinggap 0.5 >
outfile 2> errfile
```

-list *xyz* [-list, -llist]**:** Print a list of all commands that contain the string xyz. Use -list - to see all commands. See also -help, -list, -clist, -cllist, -dlist, -dllist.

-llist *xyz*[]**:** Print a list of all commands that contain the string xyz and include their usage information. Use -llist - for all commnds. See also -help, -list, -clist, -cllist, -dlist,-dllist.

-maxiterations *n* [unlimited]: Under –iterativepass, this is a stopping rule to limit the number of iterative passes to perform if stability is not achieved.

-maxprocessors *n* [number of available parallel nodes]: Number of spawned jobs in parallel (minimum n=2 for a master and a single slave).

-maxtrees *n* [hold all trees available memory will allow]: Set maximum number of trees held in buffers. The argument for this value will be used for all maxtrees commands (e.g., -buildmaxtrees n, -sprmaxtrees) unless they are specified.

-minstop *n* [0]. Minimum number of random replicates that must be completed before -stopat will come into effect. When performing –multirandom –random n, sets the minimum number of replicates that need to be done before the analysis finishes. Use in combination with –multirandom –random n and –stopat n (employed in Giribet et al., 2001).

The second command line used in the -controllers examples above can be modified to take advantage of these stopping rule commands. For example:

```
poy -parallel -controllers 8 -multirandom -random 64 -minstop 15 -stopat 3
datafile > outfile 2> errfile
```

This command line performs up to 64 full searches (build through refinement) starting by sending one to each subcluster. However unlike the original example which must go through 64 searches this run will stop when minimum tree length is found at least three times after having completed at least 15 tree searches, or after performing 64 full searches, whatever comes first. (See also –multirandom).

-molecularmatrix *matrixfile* [null] : Read matrix for molecular characters from a file. The format is five lines each with five integers signifying the transformation costs among molecular character states. Only one molecular matrix can be specified per run:

```
A->A        A->C        A->G        A->T        A->GAP
C->A        C->C        C->G        C->T        C->GAP
G->A        G->C        G->G        G->T        G->GAP
T->A        T->C        T->G        T->T        T->GAP
GAP->A      GAP->C      GAP->G      GAP->T      GAP->GAP
```

e.g., for a matrix in which gaps cost 8 transversions cost 2 and transitions cost 1 the matrix should contain:

```
0 2 1 2 8
2 0 2 1 8
1 2 0 2 8
2 1 2 0 8
8 8 8 8 0
```

-multibuild *n* [0]: Causes n random addition sequence builds (no swapping) to be performed on slave nodes. The best addition(s) are submitted to branch swapping. If -random *m* is also specified, the n builds will occur for each round of random, resulting in n x m builds and m swapping rounds. If -parallel is also specified, the builds are performed remotely. As with -random, the outgroup will be randomized unless -norandomizeoutgroup is also specified.

-multidrift [nomultidrift –nomultidrift] : Modifies the parallel behavior of -drifttbr or -driftspr by causing individual drift jobs to be spawned to slave nodes. Usage is as follows:

```
-drifttbr -multidrift -numdrifttbr n
 or
-driftspr -multidrift -numdriftspr n
```

-multiplier *n* [100]: sets weight multiplier to n. This command only has force when implied weighting is used as specified by -goloboff.

-multirandom [nomultirandom –nomultirandom] : Modifies the parallel behavior of -random by causing individual random replicates to be spawned to slave nodes. Also modified by controllers. Usage is as follows:

```
-random n -multirandom
 or
-random n -multirandom -controllers m
```

This command parallelizes full searches, sending one of n random replicates to each processor or subcluster. Multirandom is especially useful when used in combination with –minstop n –stopat n.

-multiratchet [nomultiratchet –nomultiratchet]: modifies the parallel behavior of ratcheting by causing individual ratchet jobs to be spawned to slave nodes. Usage is as follows:

```
-ratchettbr n -multiratchet
or
-ratchetspr n -multiratchet
```

-newstates *filename* [none]: Used for search-based-optimization in concert with –fixedstates to defined the state set. Each sequence inputfile (modified by –fixedstates) must have its own newstates file. They are associated in the same order as the fixedstates input files. The file

consists of a series of hypothetical ancestral sequences, separated by line feeds, and terminated by a semi-colon ";"which is often pasted in from a file in scripts, see –printhypanc. The –newstates commands need not immediately follow their associated file.

For example for one datafile:

```
poy -fixedstates datafile -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt -newstates datafile.states -compressstates > outfile 2>
errfile
```

For example for >1 datafiles:

```
poy -fixedstates datafile1 datafile2 -seed -1 -random 10 -molecularmatrix
g1tv1ts1.txt -newstates datafile1.states datafile2.states -compressstates
> outfile 2> errfile
```

-numdriftchanges *n* [20]: Number of topological changes to accept per drift round.

-numdriftspr *n* [1]: Number of spr drift rounds.

-numdrifttbr *n* [1]: Number of TBR drift rounds.

-onan [noonan –noonan]: In parallel execution, POY normally spawns jobs on slave machines only. If -onan is specified, POY will spawn jobs on the master node.

-onannum *n* [0]: Spawns n slave jobs on the master node.

-oneasis [nooneasis -nooneasis]: When using -random n -multibuild m or -multirandom –random n the addition sequence will follow the order of taxa presented in the first data file for the first build replicate.

-outgroup *taxonname* [none]: Sets the terminal taxon that will be used as outgroup for building trees by stepwise addition under –norandomizeoutgroup and –nooneasis and as outgroup for -rerootafterbuild. The taxon that is specified with this command also changes the default taxon for –plotoutgroup and –jackoutgroup.

-overwritedataprotection [on off]**:** Allow POY to overwrite inputfiles that are used in the current run (the input is read before the file is overwritten).

For example:

```
poy mydata -agree -constraint mygroups –poystrictconsensuscharfile >
mygroups -overwritedataprotection off
```

would overwrite existing constraints in file mygroups with the groups that are in the strict consensus tree of the current run.

Alternativly, the default -overwritedataprotection on checks before POY starts tree calculations and the program exits immediately if any inputfile would be overwritten (but see -overwriteprotection for shell redirection of output and write permissions).

Contrary to `-overwriteprotection`, these checks are simple lexical comparisons of filenames as entered on the commandline, taking into account the `-datadir` setting. Thus, working in the /home/me home directory of a gnu/linux box, the following command lines will prevent datafiles to be overwritten:

```
poy -mydata -molecularmatrix g1tv1ts1.txt -printtree -plotfile > 111
poy -mydata -molecularmatrix ../ g1tv1ts1.txt datadir ../ -plotfile > 111
```

But this command line one will **not** prevent datafiles to be overwritten:

```
poy -mydata molecularmatrix g1tv1ts1.txt -printtree -plotfile > ../me/111
```

See also `-poytreefile, -poybintreefile, poystrictconsensustreefile, poystrictconsensuscharfile, -printtree, -plotfile, -jackwincladfile, -phastwincladfile, -overwritedataprotection, -datadir, -molecularmatrix, -constraint`

`-overwriteprotection` `[on off]:` Protect existing files from being overwritten with outputfiles from POY. If the option is on, and an existing file would be overwritten, POY gives a notification and exits. These checks occur at the beginning of a run, so tree calculations won't even start if files would be overwritten and there is no danger of losing trees after lengthy calculations.

Contrary to `-overwritedataprotection`, `-overwriteprotection` checks physical existence of files. So if POY is run from directory /home/me on a gnu/linux box, and there is a file 'myfile' in this directory, then all of the following will make POY exit with an error message before tree calculations are started:

```
poy mydata -printtree -plotfile myfile -overwriteprotection on

poy mydata -printtree -plotfile ../../home/../home/me/myfile -
overwriteprotection on

poy mydata -printtree -plotfile ./myfile -overwriteprotection on
```

This commandline will not make POY exit (provided there is no existing myfile in `/home/me/poyoutput/`):

```
poy mydata -plotfile /home/me/poyoutput/myfile -overwriteprotection on
```

You must have write permission in the directory

Note also that these checks occur in POY, not in the shell from which you run your scripts. So, using bash standard output redirection,

```
poy mydata -overwritedataprotection on > mydata
```

would still physically transform your data into the results that you obtain with the default tree search strategy.

On the other hand:

```
poy mydata -overwritedataprotection -printtree -plotfile > mydata
```

would not because the tree calculations won't start in the first place.

See also -poytreefile, -poybintreefile, poystrictconsensustreefile, poystrictconsensuscharfile, -printtree, -plotfile, jackwincladfile, phastwincladfile, overwritedataprotection, -enabletmpfiles

-pairmatrix [nopairmatrix -nopairmatrix]: Generate pairwise distance matrix between taxa.

-parallel [noparallel –noparallel]: Execute in parallel using PVM.

-phastwincladfile *filename* [none]: writes a file with an implied alignment for the first tree in the tree buffer, in a format that is understood by Phast (Goloboff 1993-2000) and Winclada (Nixon 2002). If the file already exists, it is overwritten without any warning.

-plotencoding *encoding* [ascii]**:** Determines the characters that printtree uses to plot trees:

ascii: only ascii-characters

UTF-8**:** UTF-8 encoded unicode characters; this gives nicer output than "ascii" but requires unicode fonts and a file viewer that understands utf-8 character encoding (e.g. import as utf-8 encoded text in msword; or use vim with ":set encoding=utf-8" in a utf-8 supporting terminal (e.g. xterm –u8).

-plotechocommandline [none]**:** Echo the commandline to -plotfile at the start of -printtree output. This is turned off by default.

-plotfile *filename* [poy.tree]: Sets the filename to which printtree directs its output. If the file already exists, output is appended at the end. If it does not exist, it is created. With filename = stdout (case sensitive), output is directed to standard output.

-plotfrequencies *mode* [no]: Determines if -printtree tabulates the frequencies of clades in the best trees

no: no clade frequencies.

short: printtree tabulates majority clade frequencies (absolute and percentagewise number of occurrences) .

long: printtree tabulates frequencies of all clades (absolute and percentagewise number of occurrences) other values are interpreted as 0.

-plotinputtrees [noplotinputtrees -noplotinputtrees]: If there are inputtrees (-topology and/or -topofile), printtree will plot these trees immediately after they have been read (in addition to the regular output of -printtree at the end of the run). In combination with –nobuild this can be used to visualize trees in parenthetical notation (note that in this case it is still required to have at least one dataset).

-plotmajority *mode* [no]**:** Determines if and how printtree will plot the majority rule consensus tree.

no: majority rule consensus tree is not plotted.
short: majority rule consensus tree is plotted without clade identification numbers.
long: majority rule consensus tree is plotted with clade identification numbers (each branch is labeled with a/b, in which a is the identification number and b the percentagewise clade frequency).

-plotoutgroup *taxonname* [the first taxon in the first dataset]: Sets the single taxon that is used as outgroup for –printtree. If —outgroup taxon_x is used, -plotoutgroup defaults to taxon_x instead of the first taxon in the first data set. Useful in combination with —randomizeoutgroup (in which case the calculation of, e.g., a strict consensus tree would otherwise make little sense).

-plotstrict *mode* [long]**:** Determines if and how printtree will plot the strict consensus tree.

no: strict consensus tree is not plotted.
short: strict consensus tree is plotted without clade identification numbers.
long: strict consensus tree is plotted with clade identification numbers.

-plottrees n [1]**:** Determines if and how printtree will plot the best trees.

no: best trees are not plotted.
short: best trees are plotted without clade identification numbers.
long: best trees are plotted with clade identification numbers.

-plotwidth n [80]**:** Determines how many columns -printree uses for its output. If a tree requires more than n columns to be plotted, it is broken into subtrees of appropriate sizes. Minimum value that can be set is 25. Names of terminals that are too long (about n-5 characters) are truncated.

-**polyaddconverttorange** [polyaddconverttorange -nopolyadconverttorange]: Set how to deal with polymorphisms in additive characters when the states that occur are not contiguous. With -polyaddconverttorange, the polymorphism is converted into the smallest range that encompasses the original polymorphism (as an example, [3578] becomes [345678]). With -nopolyaddconverttorange, the whole character is turned inadditive. If this happens with -phastwincladfile specified, the data in the phastwincladfile will reflect this conversion.

See -printccode

-poyini [poyini -nopoyini]**:** Don't automatically use the file poy.ini in the current directory as a commandfile. The contradiction that arises when the command is in poy.ini itself, or in a commandfile that is reached through poy.ini, is arbitrarily resolved by assuming that in this case poy.ini contains the single comand -nopoyini. In this way the outcome is the same, whether the file is used or not or anything in between. Yet this can be overruled by explicitly entering --poy.ini in the commandline.

-poybintreefile *filename*[null]**:** Write the best tree(s) in poy parenthetical notation to file filename, with polytomies resolved and with siblings ordered (left/right) as to give correct diagnosis when the trees are re-imported later on (see -topooutgroup).

Beware of spurious resolution when using this command to create a file to export trees to tree drawing programs. In that case -poytreefile might be better. If you want graphics of your trees right away, use -printtree.

See also -ovewritedataprotection, -ovewriteprotection, poytreefile, poystrictconsensustreefile, -poystrictconsensuscharfile.

-poystrictconsensuscharfile *filename* []**:** Write the strict consensus tree as a binary and poy-readable datafile to file filename; this file can be used as a constraint file in subsequent searches.

See also -constrain, -agree, -disagree, -bremer, -overwritedataprotection, overwriteprotection, -poytreefile, -poybintreefile, -poystrictconsensustreefile.

-poystrictconsensustreefile *filename*[]**:** Write the strict consensus tree in poy parenthetical notation to file filename; see -poytreefile for possible use. Do not use this tree to optimize characters in other programs!

See also -overwritedataprotection, overwriteprotection, -poytreefile, -poybintreefile, -poystrictconsensuscharfile.

-poytreefile *filename*[]**:** Write the best tree(s) in poy parenthetical notation to file filename; this can be useful when exporting, for the purpose of preparing figures, to a program that displays trees exactly as they are read (i.e., the program does not change the resolution on the basis of some optimality criterion that it applies to character data that it may or may not have).

If you want to use the trees trees later on to rediagnose in POY, use -poybintreefile instead. If you want graphics of your trees right away, use -printtree instead.

See also -overwritedataprotection, -overwriteprotection, -poybintreefile, poystrictconsensustreefile, -poystrictconsensuscharfile.

-prealigned *filename* [noprealigned –noprealigned]: Read DNA sequences in infile with fixed alignment. Do not use gap characters ! This command is indended for loci of equal length. This can speed up searches. Also be sure to invoke –noprealigned in the command line for loci of unequal length. For example:

```
poy -prealigned datafile1 –noprealigned datafile2 -seed -1 -random 10 -
molecularmatrix g1tv1ts1.txt > outfile 2> errfile
```

-printccode [noprintccode -noprintccode]: For every next hennig86/nona datafile (untill -noprintccode), print a ccode-like summary of the characters settings in that file.

The entry for each character starts with the character number, followed by
[or] for activity (active / not active)
+ or - for additivity (additive / non-additive; additive characters have a linear character state tree)
/n with n the characer weight (taking into account -weight)
p if the characer is polymorphic (and does not have all possible states)

As an example, using a file mydata with contents

```
Xread
\'an example for entering data in hennig86/nona xread format
\'
4 5
Cichorium_intybus     0[01] 110
Rudbeckia_laciniata   01
                            101
Quercus_robur         1?    ?29
Anagallis_arvensis    02    032
;
tread
\'my inputtree\'
(3 (0 1 2))
;
cc+.-1;
ccode ] 2 /2 3.;
tread
\'more inputtrees\'
((0 1) (2 3))*
(0(1(2 3));
p/;
this is not parsed
```

the command

poy -printccode mydata

will, besides echoing the comment, print the following line to stderr:

0[+/1 1[-/1p 2]+/2 3]+/2 4]+/2

After that, the program proceeds with the default tree search strategy.

There can be multiple ccode statements in a single file and character scopes can be entered in their general form:

```
2 5   -> characters 2 and 5
2.5   -> characters 2 to 5
.5    -> characters 0 to 5
5.    -> all characters from 5 on, 5 included
.     -> all  characters
```

Ccode statements are scanned from left to right, and whatever code specifiers are currently in effect are applied to the characters in the scopes as they are read. Besides [,], +, -, and /, ccode also accepts *, which discards all previous specifiers in a single ccode statement. \'(\' and \')\', used in Pablo Goloboff\'s programs SPA and PHAST to toggle Sankoff characters, are read but ignored otherwise. The default settings are [-/1. So

```
cc /2 0.2 ] 3 4 *+ 3;
```

will turn character 4 inactive and assign weight 2 to characters 0, 1, 2, 4, and 5. Character 3 is made active but otherwise keeps its default settings [/1.

In the xread statement, the comment between single quotes is optional, but if present it must be right after the xread keyword. Next come the number of characters and the number of terminals, followed by the data, followed by a semicolon that terminates the xread statement. The data for each terminal start with the name of the terminal, followed by at least one blank character (space, tab, carriage return or linefeed) followed by the character information. Valid character codes are the digits 0 to 9; and '?' or '-' to indicate missing data. Polymorphisms must be enclosed between square brackets (such brackets are optional otherwise). Entries for different characters can but must not be separated by blanks. The file can have only one xread statement.

The inputfile is read untill a \"procedure/;\" is encountered, anything after that is not parsed. A missing \"p/\" will cause an end-of-file error. Before the first \"p/;\", the file can also have tread statements, costs statements (cf Phast), and $ statements (cg Clados). Other keywords will cause an invalid-keyword error. Commands can be abbreviated as in Hennig86 or Nona.

Costs statements and $ statements are skipped (input resumes after the next semicolon). Trees in tread statements are parsed but then ignored. If you want to use these trees as inputtrees as well, use

poy mydata -topofile mydata

In this commandline, the first occurrence of mydata sets the data to be used (not using the trees), the -topofile mydata command then instructs poy to use the trees in that file as inputtrees (skipping the data).

The optional tread comment between single brackets must follow the tread keyword. Next follow trees in parenthetical notation. Trees are separated with \'*\' (or followed by \'[n]\', with n an optional integer) and the statement is closed with a semicolon. The trees must use numerical codes for the taxa, following the order of the taxa in the xread statement that must precede the tread statement (counting starts at 0). Parentheses must be balanced and taxon codes must be separated by at least one character (blank or parenthesis). As an example, (0(1(2 3))) in the above inputfile stands for

```
 __ Cichorium_intybus
|____ Rudbeckia_laciniata
  |____ Quercus_robur
    |__ Anagallis_arvensis
```

When reading a data inputfile, poy first checks if the file starts with an xread statement (or with any other of the accepted keywords for hennig-86 like inputfiles listed above). If it does, poy assumes that the datafile is a hennig86-like inputfile and parses it accordingly. In that case, if it encounters an error in the xread statement (e.g., wrong character state code, or insufficient number of characers for a particular terminal) or in any other part of the file (e.g., a tree format error in a tread statement), it will print an error message and exit.

If, on the other hand, the file does not start as a hennig86-like file (beware of unsuported keywords like \'dread\' or \'bb\'), poy will try to read the inputfile as a sequence inputfile (see -fastashortname for accepted formats). In case it cannot parse the file as a sequence file neither, poy will mostly print two error messages and exit. The first error message (between square brackets) is the error it encountered when trying to parse the inputfile as a hennigfile, the second the error when parsing as a sequence file.

see `-fastashortname` for formatting requirements of sequence data files
see `-topology, -topofile`

`-printhypanc` `[noprinthypanc –noprinthypanc]`: Prints hypothetical ancestral sequence reconstructions (with resolved ambiguities) of the best trees to `poy.hypanc` or the filename set by `-hypancfile`.

The folllowing shell script for poy and unix utilities that can be used to 1) generate some hypothetical ancestral states 2) sort them to a minimal set of unique states and 3) use the unique states to perform searched based optimization.

```
for LOCUS in trna

do
     for REP in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

     do

    poy $LOCUS -seed -1 -random 1 -nospr -notbr -molecularmatrix
g2tv1ts1.txt -printhypanc -nospr -notbr > out.$LOCUS.tmp 2> err.$LOCUS.tmp

     cat poy.hypanc >> g2tv1ts1.$LOCUS.all.tmp

done

sort g2tv1ts1.$LOCUS.all.tmp | uniq > g2tv1ts1.$LOCUS.unq.tmp

cat g2tv1ts1.$LOCUS.unq.tmp semicolon.file > g2tv1ts1.$LOCUS.unq

rm -f *.tmp
rm -f poy.hypanc
touch poy.hypanc

poy -fixedstates $LOCUS -newstates g2tv1ts1.$LOCUS.unq -seed -1 -
random 10 -oneasis -molecularmatrix g2tv1ts1.txt > $LOCUS.sbo.out 2>
$LOCUS.sbo.err

done
```

-printlotshypanc [noprintlotshypanc —noprintlotshypanc]: Prints hypothetical ancestral sequence reconstructions (with resolved ambiguities) of intermediate and best search trees to poy.hypanc or the filename set by -hypancfile.

-printqmat [noprintqmat —noprintqmat]: Prints transition, base frequency, and other likelihood parameter information to stdout.

-printtree [noprinttree -noprinttree]: With -build (default), plot (in ascii graphics) the best trees and their strict consensus at the end of a poy run to file poy.tree. If that file exists, new output is appended at the end (a line with \'>-<\' marks the start of the -printtree output), otherwise the file is created.

With -nobuild, inputtopologies (-topofile and /or -topology) are plotted. In that case, there are two possibilities:

1. there are character datafiles. In that case, the inputtrees are diagnosed using all characters that are in use, and zero-length branches are collapsed accordingly (note that POY internally resolves polytomies of inputtopologies in an arbitrary way, which may influence diagnosis and presence of zero-length branches; it is in general not a good idea to import trees with polytomies when the trees are going to be diagnosed).

2. there are no character datafiles. In that case, the inputtrees are used with exactly the same resolution as they are read (uses temporary files, so -enabletmpfiles must be on).

Trees are considered to be unrooted, so the basal node is never resolved (there are some inconsistencies in the current version).

The default behavior of -printtree can be changed with by the options -plottrees, -plotstrict, plotmajority, -plotwidth, -plotencoding, -plotfrequencies, and -plotfile.

-qmatrix *filename* [undefined]: Read transition matrix (5 x 5) for likelihood analysis ending with a semicolon ";".

```
      A->A        A->C        A->G        A->T        A->GAP
      C->A        C->C        C->G        C->T        C->GAP
      G->A        G->C        G->G        G->T        G->GAP
      T->A        T->C        T->G        T->T        T->GAP
      GAP->A      GAP->C      GAP->G      GAP->T      GAP->GAP
;
```

e.g.

```
      0.9  0.02 0.05 0.02 0.01
      0.02 0.9  0.02 0.05 0.01
      0.05 0.02 0.9  0.02 0.01
      0.02 0.05 0.02 0.9  0.01
      0.01 0.01 0.01 0.01 0.96
```

-quick [quick -noquick]: Do not swap on non-minimal length trees during branch swapping. This forces POY to swap only on minimum length trees despite the fact that other trees have been found in the swapping process. This is analogous to "steepest descent" in PAUP (Swofford, 2001). The countercomand -noquick will make for a more exhaustive and time consuming search.

-quote *s* [not defined]: Prints any string "s" to stdout. Useful for commenting output.

-random *n* [0]: With –nojackboot, execute n random addition sequence searches (build through tree refinement as specified by defaults and user commands). This command works in concert with -multibuild m. See also -randomizeoutgroup. See –jackboot for its meaning under jackboot.

-randomizeoutgroup [randomizeoutgroup —norandomizeoutgroup]: Randomize all terminals, including the outgroup, when calculating a random addition sequence to build a tree by stepwise addition. The countercommand, **-norandomizeoutgroup** uses by default the first terminal of the first data set as the starting point for adding the other terminals (but see **-outgroup**). **-norandomizeoutgroup** should be specified for constrained runs. If —**randomizeoutgroup** is used with —**jackboot** and **-nojackstart, -jackoutgroup** should be set as well. If **-randomizeoutgroup** is used with —**printtree** to obtain consensus trees, **-plotoutgroup** should be specified as well.

-randomizeslaves [randomizeslaves **-norandomizeslaves**]: When spawning parallel jobs randomize the PVM TIDS (parallel virtual machine task identification numbers) such that jobs are dispersed somewhat evenly across a parallel cluster with multiple users.

-ratchetoverpercent *n* [**]:** Spawn out a number of extra ratchet jobs to accommodate for unequal execution time. Once a full complement of ratchet jobs are complete, unfinished jobs are killed and the search proceeds as specified under defaults or subsequent commands.

-ratchetpercent *n* [15]: Percent of characters to be reweighted in Nixon's (1999) ratchet procedure.

-ratchetseverity *n* [2]: Weight multiplier for reweighted characters characters during parsimony ratchet searches.

-ratchetslop *n* [0]: Sets slop values for ratcheting

-ratchetspr *n* [0]: Iterative rounds of ratcheting procedure using SPR.

-ratchettbr *n* [0]: Iterative rounds of ratcheting procedure using TBR.

-ratchettrees *n* [2]: Number of trees to be saved during ratchet iterations.

-recode [norecode **-norecode**]**:** Makes non-additive and additive characters faster.

-repintermediate [norepintermediate **-norepintermediate**]: print the results of individual replicates when used with **-random n or -multirandom** —**random n. See also** —**intermediate.**

-rerootafterbuild [rerootafterbuild **-nonrerootafterbuild**]: if –randomizeoutgroup is used, the trees that are obtained by stepwise addition are rerooted to the terminal that is specified with –outgroup before they are subjected to branch swapping and other tree search algorithms. Because the reconstructed ancestral states depend on the chosen outgroup (Wheeler 1996), this may change the length of trees as reported at the end of stepwise addition.

-seed *n* [-1]: Sets seed for pseudorandom number generation. An argument of -1 (**-seed -1**) will cause the system time, in seconds, to be used. A seed of 0 and higher guarantees exact reproducibility of runs under —**noparallel**. This is not the case under –**parallel**, because the program can in this case not possibly know in advance the order in which various jobs that it sends out to slaves will return their results (this depends, a.o. on the processor speed of the slave job and its

total load). With superficial tree search strategies and small tree buffers it is therefore possible that two consecutive runs of the same script give different results. This is then not a defect of the program but an indication that the script is not adequate to analyse the data at hand.

-showiterative [noshowiterative -noshowiterative]**:** Prints iterative pass progress information. Just for fun.

-slop *n* [0]: Check all cladogram lengths which are within n tenths of a percent of the current minimum value throughout the run. For example **-slop 10** checks all cladograms found within 1% of the minimum tree length. This command slows down the search but abates the heuristic inefficiency of the tree length calculations. The argument for **-slop** will be used for all "slop" values (e.g., **-checkslop; -buildslop**) unless they are specified.

-solospawn *n* [4]: Set the number of slave jobs to be spawned in a multiprocessor computer (used for stand alone machines). Use **-jobspernode n** for control of slave jobs on each slave node in a parallel cluster.

-spewbinary [spewbinary -nospewbinary]: Output binary (containing no polytomies) trees used internally in POY. You will need to use the binary trees for any cross-checking of POY results against other software.

-spr [spr –nospr]: Perform SPR branch swapping during tree search.

-sprmaxtrees *n* [hold all trees specified by **-maxtrees**]: Set the number of trees held by SPR during tree search.

-stats [stats –nostats]: Print search statistics.

-stopat *n* [do all replicates as specified by **-random n** or **–multirandom -random n**]. Sets the number of replicates that hit minimum tree length before the analysis ends. Use in combination with **–minstop** n to set a minimum number of replicates that must be run before **-stopat** will take effect (see full description above with **-minstop n**).

-submodel *s* [default is s6g user can specify s6g s10 s3g s2g s1g or s1]. Under **-likelihood** enforces symmetries in transition probabilities. S10 has each of the 10 reversible transitions able to vary (a “super-GTR”), s6g is GTR+gaps with all gaps treated equally, s3g, s2g, s1g are 3, and2 parameter models with gaps, and Jukes-Cantor models + gaps, and s1 treats all transitions as equally probable (a “super-JC”).

For example:

```
poy datafile -seed -1 -random 10 –likelhood –submodel s10 > outfile 2> errfile
```

-tbr [tbr –notbr]: Perform TBR branch swapping during tree search.

-tbrmaxtrees *n* [hold all trees as specified by **-maxtrees**]: Set the number of trees held by TBR.

-terminalsfile *fname* [none]: Use the entries in this file as names of terminals to be included in the analysis (free format: there my be multiple lines, and each line can have multiple names, or nothing at all).

There can be multiple -terminalsfiles on one command line. In that case all entries of all terminalsfiles are added. Double occurences of names are filtered out.

With -terminalsfile specified, character inputfiles and inputtrees (-topology and -topofile) will automatically be conformed to the list of names to be included (without -terminalsfile specified, any inconsistency in taxon names between inputfiles will cause the program to abort immediately).

For character datafiles, this involves two things:\

1. whenever a data file as data information for a valid terminal that is not in the list of names, that information will be skipped.
2. whenever a data file has no data information for a terminal that is in the list of names, the program will assume missing data for the sequence or set of fragments that is involved.

Other inconsistencies or errors in names of terminals in datafiles (like double occurrences of names or names that contain parentheses) will still cause the program to abort.

For each inputfile, an overview of taxa included/excluded is printed to stderr.

This feature should diminish the need to edit datafiles (you no longer need to include the names of taxa for which a sequence is missing, and you don't need to delete taxa that you don't want to include in the analysis). Without `-terminalsfile`, the old rule still applies: all inputfiles need to have exactly the same taxon sample, and missing sequences have to be explicitly indicated)

To prevent running an analysis with too much missing information, the following check is performed in each datafile: if the number of taxa to be added from the `-terminalsfile` list is more than 20% of the number of taxa that are in the file (after discounting the taxa that are NOT in `the -terminalsfile namelist`), than the program will exit. Use `-minterminals` to change this to another percentage (`-minterminals 0` will basically skip this test ; up to you).

For inputtrees (see `-topology` and `-topofile`), this involves the following:

1. whenever an inputtree has a terminal that is not in the list of names, that terminal will be pruned from the tree.
2. whenever an inputtree lacks a terminal that is in the list of names, the program will add that terminal as a basal branch to the tree.

In case trees have been modified, a list of terminals that are involved is printed to stderr. Without -terminalsfile, all inputtopologies must conform exactly to the set of taxa that are in use (this is more restrictive than in older versions of poy, where trees could have subsets of taxa). There is no -minterminals check for inputtrees. Note that, even with -terminalsfile(s) specified, individual -topology and -topofile commands must still be internally consistent (i.e. all trees in any single -topofile or -topology command must have the same taxon sample, but taxon samples may differ between -topofile and -topology comands). The check for identical inputtrees (see -toposkipidentical) occurs after the trees have been conformed to the terminals in use.

When no -terminalsfiles are specified, the set of terminals that is in use (and to which all datafiles must explicitly conform in that case) is determined as follows:

1. If there are character datafiles, the set of terminals in the first datafile on the commandline.
2. If there are no character datafiles, the set of terminals in the first -topology or -topofile command (without datasets, it is still possible to calculate consensus trees and plot trees, using -printtree).

-theta *n* [not set]: Under —likelihood sets the proportion of invariant sites, "theta."

The user can set theta for the duration of the search, for example:

```
poy datafile -seed -1 -random 10 —likelhood —theta 0.1 > outfile 2>
errfile
```

Alternatively in this example the theta will be estimated and set:

```
poy datafile -seed -1 -random 10 —likelhood invariantsitesadjust > outfile
2> errfile
```

-time [time —notime]: Displays execution time in seconds (wall clock).

-topofile *filename* [not set]:

Read cladogram topologies (single or multiple) from a file. Cladograms must be in parenthetical notation as in POY output (using full names of terminals, and with each tree followed by [n], in which n is an optional integer); the entire list of trees must be terminated with a semicolon \";\".

Alternatively, trees can be in xread/tread format, as in outputfiles that can be generated with Hennig86, Nona or TNT. In that case the topofile must have one xread statement that precedes any tread statement, and the tread statements must use numerical codes to refer to the terminals. See -printccode for more information.

Within any single -topofile, all trees must have exactly the same set of terminals. See -terminalsfile for how to deal with sets of erminals that differ between -topofiles, or with sets of termnals that differ from the terminals that are present in datasets.

Exactly what happens to the inputtrees depends on
1. whether or not character data files are present
2. the setting of -build/-nobuild (default is -build).
3. the setting of -random.

If no character datafiles are present, the setting of -build/-nobuild and -random does not matter. In that case, the trees will be read and accepted with the resolution that is provided, with the exception that the basal node will never be resolved, so (a (b (c d))) will be treated as (a b (c d)). Possible uses are, e.g., calculation of consensus trees (-printtree, -plotstrict, plotmajority, -plotfrequencies, -poystrictconsensustreefile) and constraint files

(-poystrictconsensuscharfile), a combination of several topofiles into one file with filtering out of identical trees (-toposkipidentical, -poytreefile), adding/deleting taxa from/to trees (-terminalsfile, -poytreefile), or rerooting topofiles (-topooutgroup, -poytreefile).

If character datafiles are present, the trees will be diagnosed and zero-length branches will be collapsed accordingly (if for all characters there is at least one optimization that assigns length zero to a particular branch, tha branch is collapsed - this ocasionally leads to overcollapsing in individual trees but not to loss of resolution in their strict consensus). If -nobuild is specified, these trees will further be treated as if no character data are present.

If, on the other hand, -build is on, the setting of -random matters. If -random n is specified and n > 0, poy will do n random addition sequences and add the results of these to he inputtrees before proceeding to final swapping. In the other case, poy directly proceeds to final swapping. Output of -printtree, -poytreefile, -poybintreefile, -poystrictconsensustreefile, and -poystrictconsensuscharfile occurs after final swapping.

If an outgroup taxon is specified with the command -topooutgroup, the trees will be rerooted to that outgroup.

If -toposkipidentical is specified, identical input trees (after rerooting if -topooutgroup is specified) across all -topology and -topofile commands will be filtered out.

see -topology, to read tree(s) from the command line
see -printccode for how to import xread/tread trees from Hennig86, Nona or TNT
see -terminalsfile, -printtree.

-topology "()" [null]: Input a single topology in parenthetical notation within quotation marks. The tree must be in parenthetical notation as in POY output (using full names of terminals, and an optional [n] at the end (n is an optional integer).

With -enabletmpfiles in effect, more than one tree can be specified. These multiple trees must be formatted as POY output trees (see -topofile); the xread/tread format that is allowed for -topofile is not available from the command line.

If an outgroup taxon is specified with the command -topooutgroup, the tree will be rerooted to that outgroup.

If -toposkipidentical is specified, identical input trees (after rerooting if -topooutgroup is specified) across all -topology and -topofile commands will be filtered out.

See -topofile and -printtree for interaction with -nobuild and absence/presence of character datasets.

See also -printccode -terminalsfile –topofile.

For example command lines might read:

```
poy data1.flat data2.flat data3.flat -topology "((frog chicken) insect)" >
output

poy data1.flat data2.flat data3.flat -topology "((frog chicken) insect) *
((frog insect) chicken)" > output
```

If the input topology does not contain all the taxa in the input files, the length of that subtree will be reported.

-topooutgroup *taxonname* [no]: With -topooutgroup specified, the inputtrees will be rerooted to the specified taxon immediately after they are read.

As a side effect, sibling clades in the rerooted tree are ordered from small to large (or alphabetically for terminals). This is important because the calculation of ancestral sequences is a heuristic procedure of which the result may depend upon the order in which a tree is traversed (see Wheeler 1996). Therefore, the mere reordering of siblings may lead to a change in diagnosed tree length or even in different zero-length branch assessments. If the inputtrees are trees that were obtained in a previous run (using -spewbinary) and if -topooutgroup is specified, the original tree length may therefore not be found anymore even if the inputtrees had the same outgroup.

As an example, tree (((D C) B) A) will be read and diagnosed as (((D C) B) A) when -topooutgroup is not specified, but as (A (B (C D))) when using -topooutgroup A. Even though this does not change the tree topology, the reordering of siblings can lead to different diagnosed lengths. If -topooutgroups B is specified, the tree will be read and diagnosed as (B (A (C D))).

When there are no data inputfiles, trees are not diagnosed and branches are not collapsed.

see -poybintreefile, -outgroup, -jackoutgroup, -plotoutgroup

-toposkipidentical [toposkipidentical -notoposkipidentical]: If on, filter out duplicate inputtrees. The filtering occurs after rerooting (if -topooutgroup is specified) and diagnosis + collapse of zero-length branches (if character data are present). Note that trees are considered as rooted for this check, so (a(b c)) and (b(a c)) are considered to be different trees

-totallikelihood [nototallikelihood -nototallikelihood]: Under -likelihood, this command determines the likelihood of an optimization alignment by summing all **major** optimization alignments (to sum **all**, see -trullytotallikelihood). The default is the likelihood of the single or "dominant" optimization alignment likelihood, which may be a very small fraction of the total likelihood.

-trailinggap *n* [positive integer]: Sets both leading and trailing gap cost to n. If -trailinggap is not specified, leading and trailing gaps are set equal to the overall gap cost specified by -gap n or -molecularmatrix *filename*.

-treefuse [notreefuse -notreefuse]: Performs tree fusing ala Goloboff (1999).

-treefusespr [notreefusespr -notreefusespr]: Performs SPR on new trees found during tree fusing Goloboff (1999).

-treefusetbr [notreefusetbr -notreefusetbr]: Performs TBR on new trees found during tree fusing Goloboff (1999).

-trullytotallikelihood [notrullytotallikelihood -notrullytotallikelihood]: Under -likelihood, determines the likelihood of an Optimization alignment by summing all optimization alignments. The default is the likelihood of the single or dominant optimization alignment likelihood, which may be a very small fraction of the total likelihood.

-verbose [verbose -noverbose]: Print search progress information.

-weight *n* [1]: Succeeding input file receives a weight of n.

References:

Farris J. 1988. Hennig86 v1.5; Copyright (C) J. S. Farris, Naturhistoriska riksmuseet, Stockholm, Sweden.

Farris, J., V. Albert, M. Källersjö, D. Lipscomb, and A. Kluge. 1996. Parsimony jackknifing outperforms neighbor-joining. Cladistics 12: 99-124.

Frost, D., R. Etheridge, D. Janies, and T. Titus. 2001. Total evidence, sequence alignment, and the evolution of polychrotid lizards (Squamata: Iguanania). *American Museum of Natural History, Novitates*. Number 3343.

Giribet, G., G. Edgecombe, W. Wheeler. 2001. Arthropod phylogeny based on eight molecular loci and morphology. Nature 413: 157 - 16.

Goloboff P. 1999. Analyzing large data sets in reasonable times: solutions for composite optima. Cladistics 15: 415-428.

Goloboff, P. 1993-2000. PIWE / NONA / PHAST /SPA (Parsimony with Implied Weights). Copyright (C) P. Goloboff, Instituto Miguel Lillo, Tucuman, Argentina.
http://www.cladistics.com

Janies, D and W. Wheeler. 2001. Efficiency of parallel direct optimization. Cladistics. 17: S71-S82.

Janies, D. and W. Wheeler. in press. Theory and practice of parallel direct optimization, in Techniques in Molecular Systematics and Evolution. R. Desalle, G. Giribet, and W. Wheeler (eds). Birkhauser Verlag, Basel.

Nixon, K. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. Cladistics. 15: 407-414.

Nixon, K. C. 1999. Winclada (BETA) ver. 0.9.9 PUBLISHED BY THE AUTHOR, ITHACA, NY.

Phillips, A., D. Janies, and W. Wheeler. 2000. Multiple Sequence Alignment in Phylogenetic Analysis, Molecular Phylogenetics and Evolution. 16: 317-330.

Swofford, D. L. 2001. *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods).* Version 4. Sinauer Associates, Sunderland, Massachusetts.,

Tuffley C, M. Steel 1997. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. Bull Math Biol 1997 May;59(3):581-607.

Wheeler, W. 1999. Fixed character states and the optimization of molecular sequence data. Cladistics. 15:379-386.

Wheeler, W. 1996. Optimization Alignment: The end of multiple sequence alignment in phylogenetics? Cladistics 12:1-9.

Wheeler, W. C. and C. Y. Hayashi. 1998. The phylogeny of extant chelicerate orders. Cladistics. 14: 173-192.

Wheeler W. and D. Gladstein. 1994-2000. MALIGN. Software for multiple sequence alignment. ftp://ftp.amnh.org/pub/molecular/malign