## 0.1 Pecan User Guide (6/4/2006)

Pecan is a global multiple alignment algorithm in the same vein as MLAGAN and MAVID, etc. It reads in a set of sequences in Fasta format and a Newick-tree and outputs a file containing a Multi-Fasta formatted alignment.

It can be downloaded from pages attached to the following URL.

http://www.ebi.ac.uk/ bjp/pecan/

### Example

The following command,

```
java [class path options] bp.pecan.Pecan -E '((a, b), c);' -F c.fa d.fa e.fa
```

would align the given three sequences files, 'c.fa', 'd.fa' and 'e.fa', in an output file called output.mfa. The ordering of the leaves in the tree correspond to the ordering of the sequences passed to Pecan, hence in the example shown the 'c' and 'd' sequences correspond to leaves 'a' and 'b' in the Newick tree respectively and therefore sequence 'e' to 'c'. Importantly, Pecan ignores the contents of fasta header files for this identification. Even when only two sequences are present the tree must be provided. The two flags '-E' and '-F' are needed to indicate to the parser the presence of the sequence and tree arguments. Other user arguments are discussed below.

### Dependencies

Pecan requires Exonerate, and is tested with Exonerate-1.0.0 but may work with other, or newer versions. You can download exonerate from the following URL.

http://www.ebi.ac.uk/ guy/exonerate/

Pecan itself is written in Java and should work with any Java-virtual-machine (JVM) of version 1.4.2 or higher.

### Important issues

Firstly, if you experience any problems using this program that are not easily dealt with please contact me at bjp (AT) ebi.ac.uk. If possible please send me the logging information provided by Pecan (see below) for any failed jobs.

Pecan assumes that sequences are soft-masked, that is all in upper case characters, except for repeats which should be all in lower case characters.

While it will align hard-masked (repeats as 'N' characters, everything else upper-case) and unmasked sequences you may experience problems. Any characters apart from upper and lower case 'A', 'C', 'T', 'G', 'N' and 'X' in the fasta files will cause a runtime-exception.

When speed is an issue a number of steps can be taken. Importantly, the JVM should be run in server mode for most jobs (this is achieved by calling with the -server flag), this can make as much as a sixty percent difference, in my experience. Secondly, the newest JVMs, versions 1.5 and higher, can also speed things up very significantly compared to versions 1.4x and below, mostly due to a specific issue that seems to have been fixed/improved. Choosing to use a single affine model instead of the default double affine model will also reduce run time.

## Command Line Arguments

The following is a list of command line arguments for the current version.

- -A

  Turn on logging at 'info' level. See JVM documentation on logging for further explanation.

- -B

  Set the log file (default = %t/bp.log), where %t is the default temporary directory used by the JVM.

- -C

  Set the log file logging level. Default is 'info' level.

- -D

  Set the console log level. Default is 'info' level.

- -E

  Specify the Newick-tree for the sequences, unspecified distances are given the value 1.0. A complete definition of Newick-trees can be easily found on the web. Here are a couple of examples, with and without distances on the branches.

  ((a, b), c);

  ((a:0.1, b:0.5):0.6, c:0.3);

- -F

  Sequence files in fasta format. The order must correspond to those in the tree.

- -G

  File in which to write multi-fasta alignment formatted output, default : output.mfa

- -H

  Word length of Exonerate seed k-mers. Exonerate is called recursively to lay down anchor chains, with increasingly sensitive parameters (similar to Lagan). This parameter set is used with Exonerate for recursive divide and conquer with progressively more leniant parameters, defaults : 5 8 11. See Exonerate documentation.

- -I

  Basic command upon which exonerate is run (internal parameter, mostly should not be altered), default :

  –showcigar true –showvulgar false –showalignment false

  –querytype dna –targettype dna

- -J

  Path to exonerate command, default : ‘exonerate’

- -K

  Do not perform a consistency transform between the anchor chains. T-Coffee style, chains between sequences are rescored according to projected outgroup sequences.

- -L

  Amount of edge wander to trim from anchor diagonals, default : 3. Anchor diagonals are trimmed by this amount from each end.

- -M

  Rescore alignments using relative entropy score.

- -N

  Minimum scores of Exonerate alignments for recursive divide and conquer with more leniant parameters, defaults : 100 150 200. See Exonerate documentation.

- -O

  Tell Exonerate sequences are softmasked for recursive divide and conquer with more leniant parameters, default : false true true. See Exonerate documentation.

- -P

  Exonerate saturate threshold, default : 8. See Exonerate documentation.

- -Q

  Use Exonerate gapped extension mode for recursive divide and conquer with more leniant parameters, default : false false false. See Exonerate documentation.

- -R

  Maximum distances for recursive divide and conquer with more leniant parameters, default : 20000 664000 2147483647. Avoids calling exonerate on dynamic-programming (dp) matrices greater than this many $x + y$ diagonals across. When a dp matrix is larger than the maximum it is split into two maximal matrices (one at each end), potentially leaving unaligned sequences imbetween.

- -S

  Exonerate models for recursive divide and conquer with more leniant parameters, default : affine:local affine:local affine:local. See Exonerate documentation.

- -T

  Minimum diagonal distance for calling exonerate, default : 200. Minimum length of each sequence in a dynamic programming matrix to be eligible for an exonerate call.

- -U

  Relative entropy threshold below which alignments are discarded, default : 0.65. See Chiaromonte F., Yap V.B., Miller W. 2002. Scoring pairwise genomic sequence alignments. Pacific Symposium on Biocomputing, 115-126

- -V

  Specify hmm file, default : null. Supply your own HMM in Pecan format.

- -W

  Turn off transitive anchors. Transitive anchors allow pairwise alignments to be projected between one another to restrict the area of the dynamic programming matrices.

- -X

  Width in diagonals surrounding transitive anchors, default : 10. Number of anti-diagonals (+x, -y and vice versa) to place around the transitive anchors.

- -Y

  Width in diagonals surrounding standard anchors, default : 10. Number of anti-diagonals (+x, -y and vice versa) to place around and standard anchors.

- -Z

  Size of diagonal sufficient to generate a potential cut point, default : 4. Size of diagonal for a cut point. Total length of the original exonerate diagonal will be equal to two times the edge trim plus this number.

- -a

  The minumum number of diagonal coordinates (x+y) between a cut point and the computed polygon, default : 25. Gap polygon left between cut point and computed polygon.

- -b

  The minimum number of diagonal coordinates (x+y) between a cut point and the next one, default : 500.

- -c

  Turn off consistency transformation between computed posterior probabilities. See Probcons and TCoffee for further information.

- -d

  Threshold for weights to be used in calculations, default : 0.01

- -e

  Do not use direct byte buffers. Java parameter, if a ByteBuffer exception arises you can try flipping this parameter.

- -f

  Set a minimum capacity for the weight heap (bytes), default : 1000000. Minimum size of weight heap for weights (internal parameter).

- -g

  Pre-close gaps larger than this length, default : 10000. Gaps in a sequence of greater than this length are pre-closed to avoid optimisation over very large gaps.

- -h

  Size of overhanging border (per sequence) into pre-closed gaps, default : 4500. Dynamic programming matrix at each end of pre-closed gap to allow dynamic programming in.

- -i

  Outgroup reordering diagonals distance (per sequence, internal parameter), default : 500. Experimental, avoid changing.

- -j

  Use HMM with single set of indel states. Default Pecan HMM has two sets of indel states, short and long, which helps to avoid aligning into large indels.

- -k

  Output confidence values. Basic method same as that described in the Probcons paper.

- -l

  Include not aligned probabilities in confidence values. Include the probabilities for each residue present in a column of being aligned to gap in those rows of the column containing a gap.

- -m

  Include formatted confidence values in MFA file. Allows viewing in place of confidence values.

- -n

  If confidence values are calculated, do not write out a seperate confidence values file. File contains one floating point value per line, which corrsponds to the confidence per column.

- -o

  File to write out confidence values, default : confidence.txt.

Please prefix numerical values starting with a minus (‘-’) with a ‘/’.

### HMM Input Format

Pecan has a pair-HMM input format, however until requested it will remain undocumented.

### Expectation Maximisation Training

The program bp.pecan.utils.PairEM included in the Pecan jar can be used to train pair-HMMs over very large sequences, if interested please contact me.