INDELible

Manual

This manual is not designed as a tutorial. It is a command reference guide and is meant to be something to refer to when you want to know about one command or block type. A **quicker** and **easier** way to **learn** about INDELible for the first time would be to look the through the examples in the tutorial section.

- The INDELible control file is made up of 7 different "block" types as described below.
- It does not matter what order commands are specified inside a block.
- It does not matter what order blocks are specified in the control file except that the file must begin with a [TYPE] block.
- Control files must contain at least one [MODEL], [TREE] and [PARTITIONS] block and exactly one [EVOLVE] block. There are no other restrictions.
- INDELible tries to get instructions from a file named *control.txt* which must be located in the same directory as the INDELible executable.
- If no file named *control.txt* is found then you will be prompted to give the full filename of a suitable file for INDELible to parse.
- White space (e.g. tabs, spaces and new lines) is used to separate different commands and settings in the control file so it does not matter if commands or values span multiple lines.
- When INDELible sees **//** on a line then anything else on that line is ignored and when INDELible sees **/*** all text is ignored until after the next ***/**.

More information on the commands for each block can be found by following the links.

[TYPE]

This block tells INDELible whether it is simulating under nucleotide, amino-acid or codon models and which algorithm to use.

[SETTINGS]

This block specifies non-essential user preferences such as output file types and formats, seeds for the random number generator, and whether to output detailed reports.

[MODEL]

These blocks are used to specify all aspects of the evolutionary models used during simulation. There are a variety of commands available for specifying different substitution models and indel models as well as rates of insertion and deletion to substitution etc. Codon site-models are defined here.

[TREE]

These blocks can be used to set the parameters that control random rooted and unrooted guide tree generation or to define user-specified guide trees.

[BRANCHES]

These blocks are used to simulate non-stationary and non-homogenous processes. Different models from different [MODEL] blocks can be specified on different branches of the guide-tree allowing branches to have different models of substitution, indel length distribution, rate heterogeneity, base composition etc.

[PARTITIONS]

This block type is used to set root lengths and to choose which models are used with which trees. In addition it is used when simulating multi-partitioned datasets where different sections of the simulated dataset are

created with different trees and models.

[EVOLVE]

This block tells INDELible how many replicates to generate from each [PARTITIONS] block and what the output files should be named.

INDELible

[TYPE] block

- This block tells INDELible whether it is simulating under nucleotide, amino-acid or codon models and whether to use *method 1* or *method 2* (see paper).
- Both methods give identical results but in some situations one will be faster than the other.
- It is important that this block comes first in the control file because it is used to check all subsequent input in that file for user typos.

Example Usage:

- `[TYPE] NUCLEOTIDE 1`
- `[TYPE] NUCLEOTIDE 2`
- `[TYPE] AMINOACID 1`
- `[TYPE] AMINOACID 2`
- `[TYPE] CODON 1`
- `[TYPE] CODON 2`

INDELible

[SETTINGS] block

- This block specifies non-essential user preferences such as output file types and formats, seeds for the random number generator, and whether to output detailed reports.
- If a command is not specified in the [SETTINGS] block then it will have the default value shown below in the example usage. The only exception to this is [randomseed] whose value will be randomly chosen if not specified.
- If no [SETTINGS] block is specified all commands will have these default values.
- Please click on any of the commands to learn more about them.

Example Usage:

```
[SETTINGS]
  [ancestralprint]            FALSE     // NEW, SAME or FALSE
  [output]                    PHYLIP    // FASTA, NEXUS, PHYLIP or PHYLIPT
  [phylipextension]           phy       // any alpha-numeric string
  [nexusextension]            nex       // any alpha-numeric string
  [fastaextension]            fas       // any alpha-numeric string
  [randomseed]                1568746   // any integer
  [printrates]                FALSE     // FALSE or TRUE
  [insertaslowercase]         TRUE      // FALSE or TRUE
  [markdeletedinsertions]     FALSE     // FALSE or TRUE
  [printcodonsasaminoacids]   FALSE     // FALSE or TRUE
  [fileperrep]                FALSE     // FALSE or TRUE
```

[ancestralprint]

- *NEW* will print ancestral sequences in a separate file to leaf sequences.
- *SAME* prints ancestral sequences in the same file as the leaf sequences.
- *FALSE* will not print any ancestral sequences.
- It should be noted that if you used different guide trees for different partitions in a partitioned (multi-gene) analysis then only the root sequence will be printed in SAME/NEW file specified in the last command.
- Return to example usage.

[output]

- *FASTA*, *NEXUS*, *PHYLIP* will print out sequences to file in either fasta, nexus, or phylip format respectively.
- FASTA is used by NCBI and accepted by most sequence alignment programs. NEXUS is used by e.g. MacClade, Mesquite, ModelTest, MrBayes and PAUP*. PHYLIP is used by PHYLIP and PAML.
- *PHYLIPT* gives PHYLIP format with filenames truncated to 10 characters.
- Unaligned sequences are always output in FASTA format as e.g. filename.fas
- This command sets the output type for the true alignment and prints it to a file named e.g. filename_TRUE.phy
- For more details about different output files see the examples section.
- Return to example usage.

[phylipextension]

- This command sets the file extension for true alignments in phylip format so they are e.g. filename.phy or whatever you choose them to be.
- Return to example usage.

`[nexusextension]`

- This command sets the file extension for true alignments in nexus format so they are e.g. filename.nex or whatever you choose them to be.
- Return to example usage.

`[fastaextension]`

- This command sets the file extension for fasta formatted output files so they are e.g. filename.fas or whatever you choose them to be.
- Return to example usage.

`[randomseed]`

- This must be an integer value and is used to seed the random number generator.
- Running simulations with the same random seed value (and the same control file) will produce identical datasets.
- Return to example usage.

`[printrates]`

- *TRUE* will print out a detailed output file for each replicate of each block that lists what the site-classes or relative rates of substitution are.
- *FALSE* will not print any rates information.
- Follow these links for examples of the output for NUCLEOTIDE / AMINOACID simulations, or for CODON simulations.
- Return to example usage.

`[markdeletedinsertions]`

- *TRUE* will output inserted bases/residues that have been subsequently been deleted as * rather than - for easy identification.
- *FALSE* will output all deletions as "-".
- Return to example usage.

`[insertaslowercase]`

- *TRUE* will output inserted bases/residues in lower case letters for easy identification.
- *FALSE* will output all bases/residues as upper case letters.
- Return to example usage.

`[printcodonsasaminoacids]`

- *TRUE* will output codon datasets as amino-acids - how they are translated will depend on the genetic code specified in the model.
- *FALSE* will print codons as nucleotide triplets.
- Return to example usage.

`[fileperrep]`

- *TRUE* will output each replicate dataset in a separate file.

Unaligned sequences will go in e.g. filename_1.fas, filename_2.fas, etc.
The true alignment will go in e.g. filename_TRUE_1.phy, filename_TRUE_2.phy, etc

- *FALSE* will print all replicates in a single file.
Unaligned sequences for each dataset will all go in e.g. filename.fas.
All the true alignments will go in e.g. filename_TRUE.phy
- If a file called paupstart.txt (or paupend.txt) exists in the same directory as INDELible then it will be copied to the beginning (or end) of each output file.
- If a file called paupmiddle.txt exists in the same directory as INDELible then it will be copied to e.g. filename_TRUE.phy after each replicate datset.
- These features are useful if you want to include PAUP or MrBayes blocks in your files.
- Return to example usage.

INDELible

[MODEL] block

- These blocks are used to specify the evolutionary models used during simulation.
- There are a variety of commands available for specifying different substitution models and indel models as well as rates of insertion and deletion to substitution etc.
- Each model block in the control file must be given a name (e.g. "mymodelname" below). This is used to refer to the model elsewhere in the control file.
- It does not matter whether the "white space" (between commands and values) is spaces, tabs or new lines.
- If a command is not specified the default value is used. Default values are listed in the description of each command.
- Please click on any of the commands to learn more about them.

Example Usage:

```
[MODEL] mymodelname
  [submodel]       HKY 2.5              //  HKY with kappa of 2.5
  [indelmodel]     LAV 1.7  541         //  specifies the indel length distribution
  [indelrate]      0.1                  //  rates of insertion and deletion are both 0.1
  [geneticcode]    2                    //  only used in CODON simulations
  [rates]          0.25 0.50 10         //  pinv alpha ngamcat
  [statefreq]      0.4  0.3  0.2  0.1   //  frequencies for T C A G
```

`[submodel]` for NUCLEOTIDE simulations

- Follow the links for information about this command for AMINOACID or CODON simulations.
- Most substitution models implemented in INDELible are variations of the following general matrix:

$$Q = \begin{array}{c} \\ \text{T} \\ \text{C} \\ \text{A} \\ \text{G} \end{array} \begin{array}{c} \begin{array}{cccc} \text{T} & \text{C} & \text{A} & \text{G} \end{array} \\ \begin{pmatrix} . & a\pi_C & b\pi_A & c\pi_G \\ a\pi_T & . & d\pi_A & e\pi_G \\ b\pi_T & d\pi_C & . & f\pi_G \\ c\pi_T & e\pi_C & f\pi_A & . \end{pmatrix} \end{array} \begin{array}{l} \text{TO} \\ \\ \\ \\ \text{FROM} \end{array}$$

- The different models are specified using the commands listed below in blue (the names correspond to those used by Modeltest):

N	Usage	Notes
0	[submodel] JC	a=b=c=d=e=f=1
1	[submodel] F81	a=b=c=d=e=f=1
2	[submodel] K80 a	a=f=kappa, b=c=d=e=1
3	[submodel] HKY a	a=f=kappa, b=c=d=e=1
4	[submodel] TrNef a f	a=kappa1, f=kappa2, b=c=d=e=1
5	[submodel] TrN a f	a=kappa1, f=kappa2, b=c=d=e=1
6	[submodel] K81 b c	b=e, c=d, a=f=1

7	[submodel] K81uf b c	b=e, c=d, a=f=1
8	[submodel] TIMef a b c	b=e, c=d, f=1
9	[submodel] TIM a b c	b=e, c=d, f=1
10	[submodel] TVMef b c d e	a=f=1
11	[submodel] TVM b c d e	a=f=1
12	[submodel] SYM a b c d e	f=1
13	[submodel] GTR a b c d e	f=1
14	[submodel] F84ef k	b=c=d=e=1, a=(1+k/Y), f=(1+k/R)
15	[submodel] F84 k	b=c=d=e=1, a=(1+k/Y), f=(1+k/R)
16	[submodel] UNREST TC TA TG CT CA CG AT AC AG GT GC	GA=1

N.B. *N* can be substituted for the model name.
e.g. [submodel] 0 instead of [submodel] JC

- For F84: $Y=\pi_T+\pi_C$ and $R=\pi_A+\pi_G$
- For the models with odd *N* (1-15) the base frequencies π_T, π_C, π_A, π_G are given in [statefreq].
- For the models with even *N* (0-16) the base frequencies need not be given in [statefreq] and are set automatically.
- Return to example usage.

[submodel] for AMINOACID simulations

- Follow the links for information about this command for NUCLEOTIDE or CODON simulations.
- This command is quite simple in this case. Usage is just: *[submodel] value*
- *value* is just an integer *N* or a *code* used to pick the amino-acid substitution model as defined below (references from paper given where appropriate):

N	**code**	**Reference**
0	Poisson	n/a
1	JTT	Jones et al,. 1992
2	JTT-dcmut	Kosiol and Goldman, 2005
3	Dayhoff	Dayhoff et al., 1978
4	Dayhoff-dcmut	Kosiol and Goldman, 2005
5	WAG	Whelan and Goldman, 2001
6	mtMAM	Yang et al., 1998
7	mtART	Abascal et al., 2007
8	mtREV	Adachi and Hasegawa, 1996
9	rtREV	Dimmic et al., 2002
10	cpREV	Adachi, 2000
11	Vt	Muller and Vingron, 2000
12	Blosum	Henikoff and Henikoff, 1992
13	LG	Le and Gascuel, 2008
14	HIVb	Nickle et al., 2007
15	HIVw	Nickle et al., 2007
16	USER	n/a

- For the user defined substitution model the number *16* or the code *USER* should be followed by a filename containing the rate matrix Q. For example:
  [submodel] USER userAAmodel.txt
- This file should be in the same directory as the INDELible executable.
- The formatting of this file follows the PAML convention. e.g. for the Dayhoff-dcmut model the file should be formatted like this.
- *+F* versions of these models are specified by defining stationary frequencies using the [statefreq] command.
- If the [statefreq] is not specified the stationary frequencies from the model are used..

- Return to example usage.

`[submodel]` for CODON simulations

- Follow the links for information about this command for NUCLEOTIDE or AMINOACID simulations.
- This command is [submodel] ECMunrest for the empirical unrestricted model.
- This command is [submodel] ECMrest for the empirical restricted model.
- Otherwise a codon model with K discrete omega categories is defined using the commands below in blue:

```
// M3 (discrete)                        // p(K-1)=1-p(K-2)-...-p1-p0
[submodel]  kappa
            p0  p1  ...  p(K-2)         // proportions
            ω0  ω1  ...  ω(K-2)  ω(K-1)  // omegas
```

- All other models from M0-M13 can be represented in this M3 format. For example:

```
// M0 (one-ratio)
[submodel]  kappa  &omega0                    //  p0=1

// M1 (neutral)
[submodel]  kappa  p0  ω0  1                  //  ω1=1;  p1=1-p0

// M2 (selection)
[submodel]  kappa  p0  p1  ω0  1   ω2         //  ω1=1;  p2=1-p1-p0

// M4 (freqs) with K=5
[submodel]  kappa
            p0  p1          p2          p3       //  p4=1-p3-p2-p1-p0
            0   0.333333    0.666666    1   3    //  ω0,  ω1,  ω2,  ω3,  ω4
```

- A script (named "M5-13") is provided with INDELible to calculate the discrete values for this command from the parameters used in models M5-M13.
- Return to example usage.

`[indelmodel]`

- This sets the insertion and deletion length distributions to be the same with four possible choices:

```
(1)  [indelmodel]  NB   q  r                 //  Negative Binomial Distribution
(2a) [indelmodel]  POW  a                    //  Zipfian Distribution
(2b) [indelmodel]  POW  a  M                 //  Zipfian Distribution
(3)  [indelmodel]  LAV  a  M                 //  Lavalette Distribution
(4)  [indelmodel]  USER mylengthmodel.txt    //  User-Defined Distribution
```

- (1) This specifies a Pascal (negative binomial) distribution where q is a decimal (0<=q<=1) and r is an integer (r>0).
- (2a) This specifies a Zipfian (power law) distribution where a is a decimal (a>1).
- (2b) This also specifies a Zipfian distribution where a is a decimal (a>1). However with this format indels longer than length M are not permitted. This format is highly recommended for small values of a because of the fat-tailed shape of the distribution.
- (3) This specifies a Lavalette distribution where a is a decimal (a>1) and M is an integer (M>1) representing the maximum indel length.
- (4) This specifies a user-defined indel length distribution. The file mylengthmodel.txt should be in the same directory as the INDELible executable and contain a list of relative frequencies (in order of increasing indel length) separated by white space, like this.
- If you want to specify different length distributions for insertions and deletions then do not use [indelmodel].
- In this case use the commands [insertmodel] and [deletemodel] instead. The format of these two commands is the same as for [indelmodel].
- Return to example usage.

[indelrate]

- This sets the insertion and deletion rates to both be equal to whatever value is given.
- Both rates are relative to an average substitution rate of 1.
- N.B. This means that *insertionrate* = *indelrate* and *deletionrate* = *indelrate*.
- This is not the same as *insertionrate* + *deletionrate* = *indelrate*.
- To specify different rates for insertions and deletions then do not use [indelrate].
- In this case use the commands [insertrate] and [deleterate] instead. The values for each command are the rates relative to an average substitution rate of 1.
- Return to example usage.

[geneticcode]

- This command can only be used in *CODON* simulations.
- The value should be an integer 1 to 6, 9 to 16, or 21 to 24, corresponding to the genetic codes listed on Genbank.
- The value 1 (corresponding to the universal genetic code) is the default setting if the command is not specified.
- These genetic codes determine which codons are stop codons and therefore not simulated by INDELible.
- They are also used to translate codons to amino-acids for output if that option is chosen.
- The codes listed at Genbank (in Oct. 2008) are given below (* represents a stop codon).
- Please note some codes are identical and differ only in terms of Starts. Please see Genbank for more info.

```
1  - The Standard Code
FFLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
2  - The Vertebrate Mitochondrial Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIMMTTTTNNKKSS**VVVVAAAADDEEGGGG
3  - The Yeast Mitochondrial Code
FFLLSSSSYY**CCWWTTTTPPPPHHQQRRRRIIMMTTTTNNKKSSRRVVVVAAAADDEEGGGG
4  - The Mold, Protozoan, and Coelenterate Mitochondrial
     Code and the Mycoplasma/Spiroplasma Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
5  - The Invertebrate Mitochondrial Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIMMTTTTNNKKSSSSVVVVAAAADDEEGGGG
6  - The Ciliate, Dasycladacean and Hexamita Nuclear Code
FFLLSSSSYYQQCC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
9  - The Echinoderm and Flatworm Mitochondrial Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIIMTTTTNNNKSSSSVVVVAAAADDEEGGGG
10 - The Euplotid Nuclear Code
FFLLSSSSYY**CCCWLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
11 - The Bacterial and Plant Plastid Code
FFLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
12 - The Alternative Yeast Nuclear Code
FFLLSSSSYY**CC*WLLLSPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
13 - The Ascidian Mitochondrial Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIMMTTTTNNKKSSGGVVVVAAAADDEEGGGG
14 - The Alternative Flatworm Mitochondrial Code
FFLLSSSSYYY*CCWWLLLLPPPPHHQQRRRRIIIMTTTTNNNKSSSSVVVVAAAADDEEGGGG
15 - The Blepharisma Nuclear Code
FFLLSSSSYY*QCC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
16 - The Chlorophycean Mitochondrial Code
FFLLSSSSYY*LCC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
21 - The Trematode Mitochondrial Code
FFLLSSSSYY**CCWWLLLLPPPPHHQQRRRRIIMMTTTTNNNKSSSSVVVVAAAADDEEGGGG
22 - The Scenedesmus obliquus mitochondrial Code
FFLLSS*SYY*LCC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG
23 - The Thraustochytrium Mitochondrial Code
FF*LSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG

Base1: TTTTTTTTTTTTTTTTCCCCCCCCCCCCCCCCAAAAAAAAAAAAAAAAGGGGGGGGGGGGGGGG
Base2: TTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGGTTTTCCCCAAAAGGGG
```

```
Base3: TCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAGTCAG
```

- Return to example usage.

`[rates]`

- This command has no effect in *CODON* simulations.
- The 3 entries in this command are (from left to right): *pinv*, *alpha* and *ngamcat*.
- *pinv* is the proportion of invariable sites (should be a number between 0 and 1).
- *alpha* is the shape parameter for the gamma distribution (should be a positive number).
- If *alpha*=0 then there will be no gamma rate variation.
- *ngamcat* is the number of categories to use in the discrete gamma approximation.
- If *ngamcat*=0 then continuous gamma distribution will be used for rate variation.
- Return to example usage.

`[statefreq]`

- This command is used to specify the stationary frequencies used in the model by listing them separated by white space.
- If the command is not specified in a [MODEL] block then all stationary frequencies will be set to be equal.
- If the list of numbers does not add up to 1 then they will be rescaled so that they do.
- For *NUCLEOTIDE* simulations this must be a list of 4 numbers representing the frequencies for the different nucleotides (in the order T C A G).
- For *AMINOACID* simulations this must be a list of 20 numbers representing the frequencies for the different amino-acids (in the order A R N D C Q E G H I L K M F P S T W Y V). This will change the stationary frequencies from those defined in by the substitution model specified in [submodel] (i.e. INDELible will simulate under a +F variant).
- For *CODON* simulations this must be a list of 64 numbers representing the frequencies for the different codons (in the order TTT TTC TTA TTG TCT TCC TCA .... GGA GGG).
- Care should be taken that the stationary frequencies corresponding to any stop codons in your chosen genetic code are equal to zero. If your input contradicts this then INDELible will inform you.
- Return to example usage.

INDELible

[TREE] block

These blocks are used to specify the trees used during simulation.

User Defined Trees:

- User trees can be rooted or unrooted and may contain any number of spaces, tabs or new lines.
- The tree name is used to refer to the tree in [PARTITIONS] blocks.
- The 3 user trees (named t1, t2 and t3 below) will all be read as identical.

```
[TREE] t1 ((A:0.1,B:0.1):0.1,(C:0.1,D:0.1):0.1);

[TREE] t2 ( (A:0.1, B:0.1):0.1, (C:0.1, D:0.1):0.1 );

[TREE] t3
  (             // trees can span any number of lines
    (           // and include any amount of whitespace
      A:0.1,    // including new lines.
      B:0.1
    ):0.1 ,     // comments within the tree will be ignored.
   (C:0.1,D:0.1):0.1);
```

- Any tree can also be rescaled to be a certain tree length. For example the two following trees are identical.

```
[TREE] T1 ((A:0.1,B:0.2):0.3,(C:0.4,D:0.5):0.6);
[treelength] 4.2

[TREE] T2 ((A:0.2,B:0.4):0.6,(C:0.8,D:1.0):1.2);
```

Random Trees:

- Random trees can also be created by INDELible. A different tree will be generated for each replicate.
- Random rooted and unrooted trees require the use of more commands:

```
[TREE] tree1
[unrooted] 10 2.4 1.1 0.2566 0.34  // ntaxa birth death sample mut

[TREE] tree2
[unrooted] 10 2.4 1.1 0.2566 0.34  // ntaxa birth death sample mut
[seed] 2381242

[TREE] tree3
[rooted] 15 6.7 2.5 0.234 0.31     // ntaxa birth death sample mut

[TREE] tree4
[rooted] 15 6.7 2.5 0.234 0.31     // ntaxa birth death sample mut
[treelength] 8

[TREE] tree5
[rooted] 15 6.7 2.5 0.234 0.31     // ntaxa birth death sample mut
[treedepth] 0.4
```

- tree1 and tree2 will be unrooted random trees whilst tree3 and tree4 will be rooted random trees.
- Every time that INDELible is run tree2 will produce the same sequence of random trees beginning with the

first replicate in any [EVOLVE] block where it is used, until the number after the [seed] command is changed. The other four (tree1, tree3, tree4 and tree5) will always produce different trees.

- Please note that the [seed] command overrules the use of [randomseed] in a [SETTINGS] block. If [seed] is not used, then [randomseed] will generate the same sequence of trees every time a control file is run.
- tree4 will produce random trees that are always rescaled to have a tree length of 8, and tree5 will produce random trees that are always rescaled to have a depth (root to tip) of 0.4, whilst the other three (tree1, tree2 and tree3) will always produce random trees with different tree lengths/depths. Tree Depth can only be set for random trees.
- The numbers that come after the [unrooted] and [rooted] commands are the same in both cases. The first number is the number of taxa (10 for tree1 and tree2, 15 for tree3 and tree4).
- The next four are the parameters used in the birth-death process to create the random trees.
- In order, from left to right, these are the birth-rate, death-rate, sampling fraction and mutation rate. Further details on these parameters can be found in this paper.

Branch Lengths:

- For a given topology INDELible can also create branch lengths.
- This is done by using the command [branchlengths]

```
  [TREE] EQUAL-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] EQUAL // All branch lengths will be equal
  [treedepth] 0.1       // Root-to-longest-tip distance of 0.1

[TREE] EQUAL-TREE2
  // If branch lengths are provided, they are ignored
  ((((A:0.2,B:0.1):0.4,(C:0.3,D:0.1):0.6):0.1,
  ((E:0.1,F:0.1):0.1,(G:0.2,H:0.1):0.1):0.3):0.1,
  (((I:0.1,J:0.6):0.1,(K:0.1,L:0.1):0.1):0.1,
  ((M:0.4,N:0.1):0.1,(O:0.6,P:0.1):0.1):0.1):0.1);

  [branchlengths] EQUAL // Again, all branch lengths will be equal
  [treedepth] 0.1       // Root-to-longest-tip distance of 0.1

[TREE] ULTRAMETRIC-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] ULTRAMETRIC // All branch lengths will be equal
  [treedepth] 0.1             // Root-to-longest-tip distance of 0.1

[TREE] NON-ULTRAMETRIC-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] NON-ULTRAMETRIC // All branch lengths will be equal
  [maxdistance] 0.2               // maximum pairwise distance of 0.2
```

- [treedepth] 0.1 rescales the tree to have a maximum root-to-tip distance of 0.1
- After using the [branchlengths] command you should use [treelength] or [treedepth], or [maxdistance] to rescale your tree.
- [branchlengths] EQUAL will make every branch on the tree equal to 0.1.
- [branchlengths] NON-ULTRAMETRIC gives every branch a random length between 0 and 1.
- [branchlengths] ULTRAMETRIC gives every branch a random length between 0 and 1, but will also extend the terminal branches so that the root-to-tip distance is the same for every tip.
- If the [branchlengths] command is used then the tree toplogy can be specified with or without branch lengths. It does not matter. Any branch lengths will be ignored. i.e. the trees EQUAL-TREE and EQUAL-TREE2 will be identical.
- Examples of the trees produced above can be seen here.

- All trees in the image are rescaled to have a maximum tree-depth of 0.1. This means that the root-to-tip distance for taxon G is equal to 0.1 in all 3 trees.
- N.B. For ultrametric trees [maxdistance] 0.2 is equivalent to [treedepth] 0.1. For the non-ultrametric tree they are not the same. [maxdistance] 0.2 on the non-ultrametric tree scales the tree such that the sum of the branch lengths in between (in this case) taxons G and N would be 0.2

Trees (random or user) that are used during the simulation will be output by INDELible in a file like this.

INDELible

[BRANCHES] block

These blocks are used to simulate non-stationary and non-homogenous processes. Different models from different [MODEL] blocks can be specified on different branches of the guide-tree allowing branches to have different models of substitution, indel length distribution, rate heterogeneity, base composition etc.

Example Usage:

```
[TREE]     t1 ( (A:0.1, B:0.1):0.1,(C:0.1, D:0.1):0.1);

[BRANCHES] b1 ( (A #mA, B#mB) #mAB,(C #mC, D #mD) #mCD)#mROOT;
```

- The easiest way to format a [BRANCHES] block is to cut and paste a tree from a tree block (t1 in this case) and replace the branch lengths with model names from previously defined [MODEL] blocks (mA, mB ,... etc)
- In a similar way that branch lengths are given after a ":" symbol, here model names are given after a "#" symbol.
- A model must also be added at the root of the tree also (mROOT here) and will be used in creation of the initial sequence at the root.
- Branch lengths and taxa names can be included or omitted at the user's convenience as they are ignored by INDELible - it is only the model names and their position in the parentheses that has any effect. e.g. the following four examples will produce identical results.

```
[BRANCHES] b1a ((A:0.1#mA,B:0.1#mB):0.1 #mAB, (C:0.1 #mC, D:0.1#mD):0.1 #mCD)#mROOT;

[BRANCHES] b1b ( (A #mA, B#mB) #mAB,(C #mC, D #mD) #mCD)#mROOT;

[BRANCHES] b1c ( (#mA, #mB) #mAB,(#mC,  #mD) #mCD)#mROOT;

[BRANCHES] b1d
   (
      (                        // like [TREE] blocks, [BRANCHES] trees
        A #mA, B#mB            // can contain any amount of white space
      ) #mAB                   // or comments between the first ( and
    ,                          // the ; at the end.
      (C #mC, D #mD)
    #mCD)#mROOT;
```

- For more examples please see the tutorial section.

INDELible

[PARTITIONS] block

This block type is used to set root lengths and to choose which models are used with which trees. In addition it is used when simulating multi-partitioned datasets where different sections of the simulated dataset are created with different trees and models.

Example Usage:

```
[PARTITIONS] partitionname [treename name rootlength]

[PARTITIONS] pM
  [t1  m1  rootlength1]
  [t2  m2  rootlength2]
  [t3  b3  rootlength3]
  [t4  m4  rootlength4]
  [t5  b5  rootlength5]
```

- *partitionname* is used to identify a particular [PARTITIONS] block in the [EVOLVE] block.
- *treename* is the name of any previously defined [TREE] block.
- *name* is the name of any previously defined [MODEL] or [BRANCHES] block.
- *rootlength* is the length of the root sequence that will be generated for this partition.

- If using a [BRANCHES] block instead of a [MODEL] block then remember that the trees from the [TREE] and the [BRANCHES] block must match.
- Because of this it is NOT permitted to use random trees with [BRANCHES] blocks.

- *pM* is an example of how to define the [PARTITIONS] block when you want to generate multi-partitioned datasets.
- The tree, root length and model/branch class can be different in each and every partition and there is no limit on the number of partitions other than that imposed by computer memory.
- However the trees must all have the same number of taxa and branch-classes can not be used in the same partition as random trees.

INDELible

[EVOLVE] block

This block basically defines the queue of jobs that INDELible must complete. It tells INDELible how many replicates to generate from each [PARTITIONS] block and what the output files should be named.

Example Usage:

```
[EVOLVE]
  p1 100 outputname1
  p2 100 outputname2
  p3 100 outputname3
  p4 100 outputname4
  p5 100 outputname5
  p6 100 outputname6
  p7 100 outputname7
  p8 100 outputname8
```

- *p1* etc refers to the name of previously defined [PARTITIONS] blocks.
- *100* is the number of replicates that will be generated for each job.
- *outputname1* etc will be used in the filenames of the generated datasets.
- Each control file must contain exactly **one** of these blocks.

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//  INDELible V1.03 control file - AMINOACID.txt                              //
//                                                                            //
//      An introduction to different amino-acid substitution models.          //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

  /* Again - the control file must begin with the [TYPE] statement */

[TYPE] AMINOACID 1      //  amino-acid simulation using algorithm from method 1.

  /* Many different models can be defined in a single control file */

[MODEL]    JTTexample1
  [submodel]  JTT                       //  JTT model defined using the code.

[MODEL]    JTTexample2
  [submodel]  1                         //  JTT model defined using a number.

[MODEL]    WAGexample
  [submodel]  WAG                       //  WAG defined using the code

  /* Specifying stationary frequencies will force the +F version of a model */

[MODEL]    WAG_Fexample
  [submodel]  WAG                     // WAG+F
    [statefreq]                                     // list of 20 numbers
      0.079066  0.055941  0.041977  0.053052  0.012937  // A R N D C
      0.040767  0.071586  0.057337  0.022355  0.062157  // Q E G H I
      0.099081  0.064600  0.022951  0.042302  0.044040  // L K M F P
      0.061197  0.053287  0.012066  0.034155  0.069147  // S T W Y V

  /* User defined (reversible) amino-acid substitution model */

[MODEL]   USERexample                 // e.g. formatted like this
  [submodel]  userAAmodel.txt         // for the Dayhoff-dcmut model.

  /* Many different trees can be defined in a single control file */

[TREE] t1  (A:0.1,B:0.1);
[TREE] t2  ( (A:0.1, B:0.1):0.1, (C:0.3, D:0.3):0.5 );
[TREE] t3  ( species1:0.1, species2:0.1, (species3:0.2, species4:0.2):0.01 );
[TREE] t4
 (((1:0.1,2:0.1):0.1,(3:0.1,4:0.1):0.1):0.1,((5:0.1,6:0.1):0.1,(7:0.1,8:0.1):0.1):0.1);

  /* Many different partition groupings can be defined in a single control file */

[PARTITIONS] pJTT1  [t1 JTTexample1 160]  // tree t1, model JTTexample1, root length 160
[PARTITIONS] pJTT2  [t2 JTTexample2 500]  // tree t2, model JTTexample2, root length 500
[PARTITIONS] pWAG1  [t3 WAGexample 988]   // tree t3, model WAGexample, root length 988
[PARTITIONS] pWAG2  [t4 WAG_Fexample 75]  // tree t4, model WAG_Fexample, root length 75
[PARTITIONS] pUSER  [t3 USERexample 988]  // tree t3, model USERexample, root length 988

  /* The [EVOLVE] statement is then used to list all the simulations you want to do */

[EVOLVE]
  pJTT1 40 J1out // 40 replicates generated from partition pJTT1 in file J1out.fas etc
  pJTT2 50 J2out // 50 replicates generated from partition pJTT2 in file J2out.fas etc
  pWAG1 25 Wout  // 25 replicates generated from partition pWAG1 in file Wout.fas etc
  pWAG2 10 WFout // 10 replicates generated from partition pWAG2 in file WFout.fas etc
  pUSER 13 Uout  // 13 replicates generated from partition pUSER in file Uout.fas etc
```

```
////////////////////////////////////////////////////////////////////////////////////

/*
     All amino-acid substitution models are specified as: [submodel] value
     where value is just an integer N or a code used to pick the amino-acid
     substitution model as defined below (references given where appropriate):
```

N	code	Reference
0	Poisson	n/a
1	JTT	Jones et al,. 1992
2	JTT-dcmut	Kosiol and Goldman, 2005
3	Dayhoff	Dayhoff et al., 1978
4	Dayhoff-dcmut	Kosiol and Goldman, 2005
5	WAG	Whelan and Goldman, 2001
6	mtMAM	Yang et al., 1998
7	mtART	Abascal et al., 2007
8	mtREV	Adachi and Hasegawa, 1996
9	rtREV	Dimmic et al., 2002
10	cpREV	Adachi, 2000
11	Vt	Muller and Vingron, 2000
12	Blosum	Henikoff and Henikoff, 1992
13	LG	Le and Gascuel, 2008
14	HIVb	Nickle et al., 2007
15	HIVw	Nickle et al., 2007
16	USER	n/a

```
    The only exception is the user-model where you simply specify the
    name of the file that contains the state frequencies and the rate
    matrix Q. This file should be in the same folder as INDELible.

*/
```

Please click here to return to the tutorial menu page for more examples.

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//  INDELible V1.03 control file - trees.txt                                  //
//                                                                            //
//      How to define user trees and how to generate random trees...          //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////


[TYPE] NUCLEOTIDE 1     //  nucleotide simulation using algorithm from method 1


[MODEL] mymodel  [submodel]  JC


  //  You will have already come across user-defined trees in the other examples.
  //  The following three trees are identical - "white space" of any kind is ignored.

[TREE] t1 ((A:0.1,B:0.1):0.1,(C:0.1,D:0.1):0.1);

[TREE] t2 ( (A:0.1, B:0.1):0.1, (C:0.1, D:0.1):0.1 );

[TREE] t3
    (             // trees can span any number of lines
      (           // and include any amount of whitespace
        A:0.1,    // including new lines.
        B:0.1
      ):0.1 ,     // comments within the tree will be ignored.
     (C:0.1,D:0.1):0.1);

  //  But any tree can also be rescaled to be a certain tree length.
  //  For example the two following trees are identical.

[TREE] T1 ((A:0.1,B:0.2):0.3,(C:0.4,D:0.5):0.6);
[treelength] 4.2

[TREE] T2 ((A:0.2,B:0.4):0.6,(C:0.8,D:1.0):1.2);


   //  Random rooted & unrooted trees can also be created by INDELible.
   //  A different tree will be generated for each replicate.
   //  Explanation of these commands follow the examples...

[TREE] tree1
  [unrooted] 10 2.4 1.1 0.2566 0.34  // ntaxa birth death sample mut

[TREE] tree2
  [unrooted] 10 2.4 1.1 0.2566 0.34  // ntaxa birth death sample mut
  [seed] 2381242

[TREE] tree3
  [rooted] 15 6.7 2.5 0.234 0.31     // ntaxa birth death sample mut
  [treedepth] 0.4

[TREE] tree4
  [rooted] 15 6.7 2.5 0.234 0.31     // ntaxa birth death sample mut
  [treelength] 8

/*
  * Random trees always have taxa names that are numbers.
    e.g. (1:0.1,2:0.1,(3:0.1,4:0.1):0.1);
```

```
  * tree1 and tree2 will be unrooted random trees.
  * tree3 and tree4 will be rooted random trees.

  * Every time that INDELible is run tree2 will produce the same sequence
    of random trees, until the number after the [seed] command is changed.
  * The others (tree1, tree3 and tree4) will always produce different trees.
  * Please note that the [seed] command overrules the [randomseed]
    command from a [SETTINGS] block, for a random tree.

  * tree4 will produce random trees that are always rescaled to have a tree
    length of 8, and tree3 will produce random trees that are always rescaled
    to have a depth (root to tip) of 0.4, whilst the other two (tree1 and
    tree2) will always produce random trees with different tree lengths/depths.

  * The numbers that come after the [unrooted] and [rooted] commands are
    the same in both cases.  The first number is the number of taxa
    (10 for tree1 and tree2, 15 for tree3 and tree4).
  * The next  four are the parameters used in the birth-death process to
    create the random trees. In order, from left to right, these are the
    birth-rate, death-rate, sampling fraction and mutation rate.
    Further details on these parameters can be found in this paper:
    http://abacus.gene.ucl.ac.uk/ziheng/pdf/1997YangRannalaMBEv14p717.pdf

  * Trees (random or user) that are used during the simulation will be output
    by INDELible in a file like this.

  * One last point to remember about random trees is that they CANNOT be
    used when you want the evolutionary model to be permitted to change
    over the tree. i.e. they CANNOT be used with [BRANCHES] blocks.

*/


   //  For a given topology INDELible can also create branch lengths.
   //  This is done by using the command [branchlengths]
   //  Further explanation of the four examples is given below



[TREE] EQUAL-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] EQUAL // All branch lengths will be equal
  [treedepth] 0.1       // Root-to-longest-tip distance of 0.1

[TREE] EQUAL-TREE2
  // If branch lengths are provided, they are ignored
  ((((A:0.2,B:0.1):0.4,(C:0.3,D:0.1):0.6):0.1,
  ((E:0.1,F:0.1):0.1,(G:0.2,H:0.1):0.1):0.3):0.1,
  (((I:0.1,J:0.6):0.1,(K:0.1,L:0.1):0.1):0.1,
  ((M:0.4,N:0.1):0.1,(O:0.6,P:0.1):0.1):0.1):0.1);

  [branchlengths] EQUAL // Again, all branch lengths will be equal
  [treedepth] 0.1       // Root-to-longest-tip distance of 0.1

[TREE] ULTRAMETRIC-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] ULTRAMETRIC // All branch lengths will be equal
  [treedepth] 0.1             // Root-to-longest-tip distance of 0.1

[TREE] NON-ULTRAMETRIC-TREE
  // No branch lengths need to be provided
  ((((A,B),(C,D)),((E,F),(G,H))),(((I,J),(K,L)),((M,N),(O,P))));

  [branchlengths] NON-ULTRAMETRIC // All branch lengths will be equal
  [maxdistance] 0.2               // Maximum pairwise distance of 0.2
```

```
/*
  * [treedepth] 0.1 rescales the tree to have a maximum root-to-tip distance of 0.1
  * After using the [branchlengths] command you should use [treelength] or
    [treedepth], or [maxdistance] to rescale your tree.

  * [branchlengths] EQUAL will make every branch on the tree equal to 0.1.
  * [branchlengths] NON-ULTRAMETRIC gives every branch a random length between 0 and 1.
  * [branchlengths] ULTRAMETRIC gives every branch a random length between 0 and 1,
    but will also extend the terminal branches so that the root-to-tip distance is
    the same for every tip.

  * If the [branchlengths] command is used then the tree toplogy can be specified
    with or without branch lengths. It does not matter. Any branch lengths will be ignored.
    i.e. the trees EQUAL-TREE and EQUAL-TREE2 will be identical.

  * Examples of the trees produced above can be seen here.

  * All trees in the image are rescaled to have a maximum tree-depth of 0.1.
    This means that the root-to-tip distance for taxon G is equal to 0.1 in all 3 trees.

  * N.B. For ultrametric trees [maxdistance] 0.2 is equivalent to [treedepth] 0.1
    For the non-ultrametric tree they are not the same.
    [maxdistance] 0.2 on the non-ultrametric tree scales the tree such that the sum of
    the branch lengths in between (in this case) taxons G and N would be 0.2

*/


[PARTITIONS] Pname1  [t1 mymodel 1000]   // tree t1, model mymodel, root length 1000
[PARTITIONS] Pname2  [t2 mymodel 1000]   // tree t2, model mymodel, root length 1000
[PARTITIONS] Pname3  [t3 mymodel 1000]   // tree t3, model mymodel, root length 1000
[PARTITIONS] Pname4  [T1 mymodel 1000]   // tree T1, model mymodel, root length 1000
[PARTITIONS] Pname5  [T2 mymodel 1000]   // tree T2, model mymodel, root length 1000



[EVOLVE]     Pname1  10  outputname1     // 10 replicates generated from partition Pname1
             Pname2  10  outputname2     // 10 replicates generated from partition Pname2
             Pname3  10  outputname3     // 10 replicates generated from partition Pname3
             Pname4  10  outputname4     // 10 replicates generated from partition Pname4
             Pname5  10  outputname5     // 10 replicates generated from partition Pname5
```

Please click here to return to the tutorial menu page for more examples.

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//  INDELible V1.03 control file - settings.txt                               //
//                                                                            //
//      How to include paup blocks, set seeds for the random number generator,//
//      change the output format for generated data, or how to print "reports"//
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

[TYPE] NUCLEOTIDE 2     //  nucleotide simulation using algorithm from method 2

  //
  // * If a command is not specified in the [SETTINGS] block then it will have the
  //   default value shown below. The only exception to this is [randomseed] whose
  //   value will be randomly chosen if not specified.
  // * If no [SETTINGS] block is specified all commands will have these default values.
  // * More information on each of the commands can be found at the end of the file.
  //


[SETTINGS]
  [ancestralprint]            FALSE     // NEW, SAME or FALSE
  [output]                    PHYLIP    // FASTA, NEXUS, PHYLIP or PHYLIPT
  [phylipextension]           phy       // any alpha-numeric string
  [nexusextension]            nex       // any alpha-numeric string
  [fastaextension]            fas       // any alpha-numeric string
  [randomseed]                1568746   // any integer
  [printrates]                FALSE     // FALSE or TRUE
  [insertaslowercase]         TRUE      // FALSE or TRUE
  [markdeletedinsertions]     FALSE     // FALSE or TRUE
  [printcodonsasaminoacids]   FALSE     // FALSE or TRUE
  [fileperrep]                FALSE     // FALSE or TRUE


[MODEL] mymodel  [submodel]  JC


[TREE] t1 ((A:0.1,B:0.1):0.1,(C:0.1,D:0.1):0.1);


[PARTITIONS] Pname  [t1 mymodel 1000]   // tree t1, model mymodel, root length 1000


[EVOLVE]     Pname  10  filename        // 10 replicates generated from partition Pname

/*
  [ancestralprint]
  * NEW will print ancestral sequences in a separate file to leaf sequences.
  * SAME prints ancestral sequences in the same file as the leaf sequences.
  * FALSE will not print any ancestral sequences.
  * It should be noted that if you used different guide trees for different partitions
    in a partitioned (multi-gene) analysis then only the root sequence will be printed
    in the SAME/NEW file specified in this command.

  [output]
  * Unaligned sequences are always output in FASTA format as e.g. filename.fas
  * This command sets the output type for the true alignment and prints it to a file
    named e.g. filename_TRUE.phy
  * FASTA, NEXUS, PHYLIP will print out sequences to file in either
    fasta, nexus, or phylip format respectively.
  * FASTA is used by NCBI and accepted by most sequence alignment programs.
    NEXUS is used by e.g. MacClade, Mesquite, ModelTest, MrBayes and PAUP*.
```

```
    PHYLIP is used by PHYLIP and PAML.
  * PHYLIPT gives PHYLIP format with filenames truncated to 10 characters.

  [phylipextension]
  * This command sets the file extension for true alignments in phylip format so they
    are e.g. filename.phy or whatever you choose them to be.

  [nexusextension]
  * This command sets the file extension for true alignments in nexus format so they
    are e.g. filename.nex or whatever you choose them to be.

  [fastaextension]
  * This command sets the file extension for fasta formatted output files so they
    are e.g. filename.fas or whatever you choose them to be.

  [randomseed]
  * This must be an integer value and is used to seed the random number generator.
  * Running simulations with the same random seed value (and the same control file)
    will produce identical datasets.

  [printrates]
  * TRUE will print out a detailed output file for each replicate of each block
    that lists what the site-classes or relative rates of substitution are.
  * FALSE will not print any rates information.
  * Follow these links for examples of the output for NUCLEOTIDE / AMINOACID simulations,
    or for CODON simulations.

  [markdeletedinsertions]
  * TRUE will output inserted bases/residues that have been subsequently been
    deleted as * rather than - for easy identification.
  * FALSE will leave them as -.

  [insertaslowercase]
  * TRUE will output inserted bases/residues as lower case letters for easy identification.
  * FALSE will output all bases/residues as upper case letters.

  [printcodonsasaminoacids]
  * TRUE will output codon datasets as amino-acids - how they are translated will
    depend on the genetic code specified in the model.
  * FALSE will print codons as nucleotide triplets.

  [fileperrep]
  * TRUE will output each replicate dataset in a separate file.
    Unaligned sequences will go in e.g. filename_1.fas, filename_2.fas, etc
    The true alignment will go in e.g. filename_TRUE_1.phy, filename_TRUE_2.phy, etc
  * FALSE will print all replicates in a single file.
    Unaligned sequences for each dataset will all go in e.g. filename.fas
    All the true alignments will go in e.g. filename_TRUE.phy
  * If a file called paupstart.txt (or paupend.txt) exists in the same directory as
    INDELible then it will be copied to the beginning (or end) of each file.
  * If a file called paupmiddle.txt exists in the same directory as INDELible then it
    will be copied to e.g. filename_TRUE.phy after each replicate datset.
  * These features are useful if you want to include PAUP or MrBayes blocks in your files.
*/
```

Please click here to return to the tutorial menu page for more examples.

```
////////////////////////////////////////////////////////////////////////////////
//                                                                            //
//  INDELible V1.03 control file - basicaminoacid.txt                         //
//                                                                            //
//      A basic introduction to the structure of the INDELible control file.  //
//                                                                            //
////////////////////////////////////////////////////////////////////////////////

// It is useful to know that anything on a line after two forward slashes is ignored.

/*
   Another useful thing to know is that anything after a forward slash and star
   is ignored until INDELible sees a star followed by a forward slash later on.
*/

[TYPE] AMINOACID 2      //  EVERY control file must begin with a [TYPE] command.
                        //  The number after "AMINOACID" can be 1 or 2 and chooses the
                        //  algorithm that INDELible uses (see manuscript). Both give
                        //  identical results but in some cases one is quicker.
                        //  Other blocks and commands following this statement
                        //  can come in any order you like.

[MODEL]    modelname          //  Evolutionary models are defined in [MODEL] blocks.
  [submodel] WAG              //  Here the substitution model is simply set as WAG.
  [indelmodel]  POW  1.7 500 //  Power law insertion/deletion length distribution (a=1.7)
  [indelrate]   0.1          //  insertion rate = deletion rate = 0.1
                             //  relative to average substitution rate of 1.

[TREE] treename  (A:0.1,B:0.1);        //  User trees are defined here

[PARTITIONS] partitionname             //  [PARTITIONS] blocks say which models go with
  [treename modelname 1000]            //  which trees and define the length of the
                                       //  sequence generated at the root (1000 here).

[EVOLVE] partitionname 100 outputname  //  This will generate 100 replicate datasets
                                       //  from the [PARTITIONS] block named above.

// The true alignment will be output in a file named outputname_TRUE.phy
// The unaligned sequences will be output in a file named outputname.fas
// To learn how to implement more complicated simulations (or different
// models) please consult the manual or the other example control files.
```

Please click here to return to the tutorial menu page for more examples.