# indel-Seq-Gen v2.0 Manual

Cory L. Strope*, Computer Science and Engineering,
Kevin Abel, Computer Science and Engineering,
Stephen D. Scott, Computer Science and Engineering,
Etsuko N. Moriyama, School of Biological Sciences and the Center for Plant Science Innovation

University of Nebraska – Lincoln

November 20, 2008

## Contents

**1 Acknowledgments** **3**

**2 Overview** **4**

**3 Getting Started** **4**

**4 iSGv2 Parameters** **5**
4.1 Global Options . . . 5
4.1.1 Discrete steps and Trace file . . . 8
4.2 Partition Options . . . 8
4.3 Lineage-specific and Motif Options . . . 10
4.3.1 Specifying Subtrees in Newick Format . . . 11
4.3.2 LINEAGES . . . 11
4.3.3 MOTIFS . . . 12

**5 Root Sequence Input** **14**
5.1 Root Sequence Input . . . 14
5.2 Multiple Alignment Input . . . 15
5.2.1 Multiple alignment root sequence input for template motifs . . . 17
5.3 Invariable Array . . . 17

**6 Other Input Files** **18**
6.1 Character Frequencies `#f`*freq_file*`#` . . . 18
6.2 Indel Probabilities {`#`$_1$`,#`$_2$`,`*indel_file(s)*} . . . 18

**7 Examples** **18**
7.1 Basic coding and non-coding . . . 18
7.2 Including indels . . . 19
7.3 Coding and non-coding together . . . 19
7.4 Lineage Introduction . . . 20

*Corresponding Author, University of Nebraska, email: corystrope@gmail.com

7.5 Prosite-like motif . . . . . 20
7.6 Template motif . . . . . 24

# 1 Acknowledgments

We would like to thank the authors of Seq-Gen, Drs. Andrew Rambaut and Nick Grassly, which provided the starting point for the development of iSG. As such, when citing indel-Seq-Gen, please note that Seq-Gen [3] should also be cited. The bibtex citation is:

```
@ARTICLE{Rambaut97,
        AUTHOR = "A. Rambaut and N.C. Grassly",
        TITLE = "{Seq-Gen}: an application for the {Monte Carlo} simulation of
{DNA} sequence evolution along phylogenetic trees",
        JOURNAL = "{CABIOS}",
        VOLUME = 13,
        NUMBER = 3,
        YEAR = 1997,
        PAGES = "235-238"       }
```

We are also grateful for the NSF AToL grant 0732863 and Department of Education grant P200A040150 for funding this work.

# 2 Overview

indel-Seq-Gen is a tool to simulate the evolutionary events of highly diverged DNA (coding and non-coding) and protein sequences. Long-term evolution often include dynamic changes such as insertions and deletions (indels), but some subsequences, such as domains and motifs, retain their original sequences and structures better than less functionally important regions. indel-Seq-Gen allows for the simulation of many different evolutionary patterns over different regions of a sequence, in the end outputting the "true" multiple alignment of the sequences. indel-Seq-Gen also has multiple unique features, including input a multiple alignment for the root sequence[1], placing functional constraints on sequence sites, tracing mutations along the simulation and noting their positions in the true multiple alignment, changing evolution parameters and site conservation between subtrees (i.e., lineages), and more. In addition iSGv2 fixes a fundamental flaw in representing in representing insertions and deletions through the introduction of evolving along the guide tree in discrete steps. Finally, iSGv2 also introduces a novel method of representing conservation patterns with respect to insertions versus deletions versus substitutions. These features can be used in many evolutionary studies, such as to test the accuracy of multiple alignment methods, phylogenetic methods, evolutionary hypotheses, ancestral sequence reconstruction methods, and superfamily classification methods.

# 3 Getting Started

indel-Seq-Gen version 2.0 (iSGv2) is freely downloadable at `http://bioinfolab.unl.edu/~cstrope/iSG/`. iSGv2 has been tested MacOS X (versions 10.4.7 and Leopard). You will need `g++` compiler if the provided executables do not run on your system.

1. Download the `indel-seq-gen-2.0.tar.gz` source archive, open the archive by typing the two commands:

   `gunzip indel-seq-gen-2.0.tar.gz`, and then

   `tar -xvf indel-seq-gen-2.0.tar`

2. This creates a directory `indel-seq-gen-2.0` with all of the files. Go to this directory `cd indel-seq-gen-2.0`. Make the executables of indel-seq-gen by typing the command "`./configure`", followed by the command "`make`". For further installation instructions, refer to the file `INSTALL`.

3. After making the executables, type the command "`cp src/indel-seq-gen data/`". This copies the executable into the `data` directory.

4. Go to the `data` directory using the command "`cd data/`".

5. In the `data` directory, there are sample files that can be used to run indel-seq-gen. The following commands will execute indel-seq-gen using various capabilities. Cut-and-paste (or type) these examples to run the simulations. After each run, you can examine the output of these runs by opening the filenames that begin with the name following the `--outfile` or `-e` flags:

```
./indel-seq-gen --matrix HKY --outfile DNA_out < simple_nuc.tree
./indel-seq-gen --matrix HKY --codon_rates 0.2,0.05,0.75 --outfile DNA_out < simple_nuc.tree
./indel-seq-gen -m HKY -e mid_noncoding --num_runs 5 < mid_nuc_noncoding.tree
./indel-seq-gen -m HKY -e mid_noncoding -n 5 -c 2,1,8 --invar 0.02 < mid_nuc_coding.tree
./indel-seq-gen -m HKY -c 2,1,8 -e exon_intron --step_type trs < exon_intron.tree
./indel-seq-gen -m HKY --lineage exon_intron_lineage.spec --alpha 1.3 < exon_intron_lineage.tree
./indel-seq-gen -m JTT -k lipocalin.spec -w a < lipocalin.tree
./indel-seq-gen -m JTT -k lipocalin_ma.spec < lipocalin_ma.tree
```

   For further understanding of these examples, refer to Section 7.

[1] This can be used to create different but related ancestral sequences in each run

# 4 iSGv2 Parameters

Creating realistic sequences requires many parameters. Because of this, iSGv2 has many options, both for the global simulation run and for subsequences, called *partitions*.

## 4.1 Global Options

To run indel-Seq-Gen from the command prompt, type the following line:

```
./indel-seq-gen -m <matrix> [-options] < tree_file > outfile
```

The "tree_file" is described in Section 4.2. "> `outfile`" is to save all the messages `indel-Seq-Gen` generates in the file. If "> `outfile`" is not added to the command, everything will be simply displayed on the screen (but not saved in any file).

`indel-Seq-Gen` has many options. We suggest that you first run some of the examples shown in Section 7. All necessary files are included in the provided `indel-seq-gen-2.0` directory. If you get any error message and the program does not run, please make sure if you followed the steps explained in the previous section. If you still cannot run the program, please contact us. The contact information is found in the first page.

Table 1: Global options (entered at the command line) and their effects for the indel-Seq-Gen run. Subsequence options (described in the next Section) will override the global options if there are conflicts. For input type {*list*}, **do not** use spaces to separate list items.

| Option | Long Option | Type | Effect |
|---|---|---|---|
| -a | --alpha | {*float*} | Shape (alpha) for gamma rate heterogeneity. |
| -b | --option_width | {*int*} | The number of residues per line on the multiple alignment output [default = 60]. |
| -c | --codon_rates | {*list*} | #1, #2, #3 = rates for codon position heterogeneity. Example: -c 0.15,0.05,0.8. Numbers are not required to sum to 1, as iSGv2 will normalize them. |
| -d | --tree_scale | {*+float*} | Total tree scale, multiplies each branch length by scale / tree length [default = tree length]. |
| -e | --outfile | {*string*} | Filename for output. Files <filename>.root, <filename>.seq, <filename>.ma, <filename>.trace and <filename.verb> will be created holding the root sequences, sequence files, multiple alignments, traced events, and verbose output from the run. |

*Continued on next page...*

Table 1: (continued)

| Option | Long Option | Type | Effect |
|---|---|---|---|
| -f | --frequencies | {*list*} | amino acid (20, ARNDCQEGHILKMFPSTWYV) or nucleotide (4, ACGT) frequencies, separated by commas, or e to use equal frequencies (0.05 or 0.25 for each state, respectively) [default = use frequencies based on the substitution matrix]. Example: -f 0.2,0.2,0.3,0.3 for 20% A and 20% C. |
| -g | --num_gamma_cats | {*int*} | Number of categories for the discrete gamma-distribution rate heterogeneity. Must be between 2 and 32 [default = none]. Higher numbers of discrete categories severely affect the speed of iSGv2. |
| -h | --help | | Output the usage instructions |
| -i | --invar | {*float*} | Proportion of invariable sites [default = none]. |
| -k | --lineage | {*filename* } | Subtree specification file name. See Section 4.3. |
| -j | --step_type | {*des, trs*} | des = discrete evolutionary steps [default], trs = time relative steps. These are described below, in Section 4.1.1. |
| -l | --length | {*int*} | Deprecated. This is no longer necessary to input. Sequence length. |
| -m | --matrix | {*string*} | Substitution matrix: HKY, F84, GTR for nucleotides, PAM, JTT, MTREV, CPREV, GENERAL for amino acids. |
| -n | --num_runs | {*int*} | The number of datasets to simulate for each tree [default = 1]. |
| -o | --outfile_format | {*char*} | Output format: either Phylip (p), NEXUS (n), or FASTA (f) [default = Phylip]. |
| -q | --quiet | | Quiet mode: only the root sequence, resultant sequences, and multiple alignment are printed. |
| -r | --rel_rates | {*list*} | Six comma-separated numbers specifying general rate matrix. |
| -s | --branch_scale | {*float*} | Branch scale factor [default = 1.0]. |
| -t | --tstv | {*float*} | Transition-transversion ratio. |

*Continued on next page...*

Table 1: (continued)

| Option | Long Option | Type | Effect |
|---|---|---|---|
| -u | --indel_fill | {*string*} | Indel fill model, based on neighbor effects [5]: xia = original from [5], built on the E. coli K-12 proteins, sp = swiss-prot, ran = no neighbor effect [default = ran]. |
| -w | --write_anc | | Write ancestral sequences. |
| -z | --rng_seed | {*int*} | Manually set the random number seed. |

#### 4.1.1 Discrete steps and Trace file

One of the most radical changes introduced by iSGv2 is the idea of using discrete steps during the evolution process. The rationale is twofold: (i) There is currently no continuous model for indel evolution (refer to the as yet unwritten paper for the reasoning), and thus most current simulation methods simulate with a flawed indel representation, and (ii) discrete steps allow for concrete tracking of events during the simulation, as described below.

**Discrete Evolutionary Steps:** The DES model simply calculates $\epsilon$, and simulates sequences along the guide tree in $\epsilon$ step sizes. It is also possible to relate event occurrence times along the simulation guide tree (called the *time relative steps*, or TRS, model), as long as two assumptions are met: (1) The input guide trees cover a single period of time, such that the roots of the guide trees (i.e., partitions) start at one time point, and the set of all taxa end at the same time, and (2) mutations occur uniformly at random throughout the simulation run. If these are true, iSGv2 scales the branch lengths of the guide trees and branch-specific $\epsilon$'s so that all mutation events that occur in simulation run are ordered with respect to the relative time of the run (i.e., the simulation starts at relative time point 0.0, and all taxa end at time point 1.0).

**Event Tracking**

A benefit introduced by the discrete step method is the ability to track indel and substitution events by:

1. ID
2. Time of occurrance (for the TRS model),
3. Set of taxa affected by change (i.e., subtree of occurrence),
4. Event type (substitution/insertion/deletion),
5. Indel length, and
6. Positions in the MSA that reflect the event.

The time of occurrance for events is listed in different ways depending on whether the user is using DES or TRS. For DES, events are listed by partition, whereas for TRS, events are reported as an ordered list based on the relative time that they occurred. As an example, observe the multiple alignment in Figure 7. Assuming a phylogeny that groups (Taxon1,Taxon2) and (Taxon3,Taxon4), positions 16–18 could have occurred in two ways, shown in Figure 1.

### 4.2 Partition Options

Specifications for each partition are given in the `tree_file`. Specifications for each subsequence are separated by ';'. Refer to the Examples section (Section 7) for some sample tree files.

Partition options allow the user to create subsequence-specific effects by overriding the global parameters that are set as in Table 1. These options are shown in Table 2.

Table 2: Subsequence options.

| Options | Command Styles | Description |
|---|---|---|
| *label* | "Partition Label" | This option, when present, will label the boundaries of the subsequence in the multiple alignment in the `*.verb` file with the name given. |

*Continued on next page...*

Table 2: (continued)

| Options | Command Styles | Description |
|---|---|---|
| *subseq* | #i<% invariable>,<br>b<branch scale>,<br>f<AA frequency file>,<br>m<substitution matrix>,<br>r# | These options modify (b) or replace (i,f,m,r) the global options given at the command line. Option b will modify the default branch length scale by the value given. Option r is a flag that specifies that no rates are used for this partition. This is helpful for specifying introns in coding region sequences, as shown in 7.3. Multiple options should be separated by commas. |
| *rootseq* | [:<root_sequence_file> ] or<br>[:<root_sequence_file>, #] or<br>[:<mult_align_file> $(1, 2, 3)$] or<br>[:<mult_align_file> $(1, 2, 3)$, #] or<br>[length] or<br>[length, #] | This option specifies the root sequence parameters. The first two formats specify the root sequence file. The third and fourth format specify the multiple alignment file, where '1' is the range of the multiple alignment to use, '2' is the number of sequences to select from the alignment, and '3' is the method of creating the consensus root sequence. For further details, see Section 5. The last two formats, a numeric value (not preceded by ':'), is to generate a random sequence of the given length and use as the root sequence. When using the options with '#', relative rates will be assigned to each partition. Using the '#' option should be done only when the trees for each partition are the same. |
| *indel* | Format:<br>{$\#_1$, $\#_2$, <*file*$_1$>/<*file*$_2$>}<br>$\#_1$: Max indel size<br>$\#_2$: Indel probability distribution<br>If $\#_2 = 0$: Use Chang & Benner<br>If $\#_2 > 0$: P(ins)=P(del)=$\#_2$<br>If $\#_2/\#_3$: P(ins)=$\#_2$, P(del)=$\#_3$ | These options specify the different indel models and parameters. Only $\#_1$ and $\#_2$ are required. <file$_1$>/<file$_2$> can be used to specify the two file names, which provide the user defined insertion length distribution (file$_1$) and deletion length distribution (file$_2$). If no distribution file is provided, the distributions given by Chang and Benner (2004) will be used[2]. If only one distribution file is given, it will be used for both insertions and deletions. |
| tree | Newick Format | The rooted evolutionary tree for this subsequence. For all subsequences, their trees must have the same number of taxa, with each taxon named the same in all trees. The branching pattern as well as branch lengths, however, may vary. Branch lengths are assumed to be the expected number of substitutions per site. **Taxon names should not begin with a number**. |

[2]For the moment, both DNA and protein. Future versions may include a DNA-specific model

```
bit(0) = Taxon1
bit(1) = Taxon2
bit(3) = Taxon3
bit(4) = Taxon4
```

```
[X,I,,1100,3,16:17:18]
```

OR

```
[X,D,,0011,3,16:17:18]
```

OR

```
[X,D,,0011,1,17]
[Y,I,,1100,2,16:18]
```

Figure 1: Three different ways that the events in Figure 7 could have occurred, as shown by the trace file using the discrete evolutionary steps (des) paradigm, where `X` is the unique event number describing the change. Since this is using des, note that the relative time field is left empty.

### 4.3 Lineage-specific and Motif Options

Lineages and motifs are input to iSGv2 through the use of the global option "`-k`" or "`--lineage`". This file consists of two parts:

1. LINEAGES, which change the conditions under which a subtree evolves, and
2. MOTIFS, which impose site-specific functional constraints on specific sequence positions for a specific subtree.

The basic outline of this file is:

```
LINEAGES =
{
    subtree_list: "lineage_name"#subseq#{indel};
    .
    .
    .
}

MOTIFS =
{
    subtree:
        MARKER=<marker>;
        NAME=<motif_description>;
        PATTERN=<modified_PROSITE_motif>;
    .
    .
    .
}
```

Lineages and motifs both work along subtrees of the input guide tree. To make a lineage or motif effective for the entire guide tree, set `subtree:` to `root:`. In order to specify subtrees, however, it is required to label clades in the Newick Format, as discussed below.

```
(
 (
  (Taxon1:0.1,Taxon2:0.1
  )Clade2:0.1,Taxon3:0.1
 )Clade1:0.1,
 (
  (Taxon4:0.1,Taxon5:0.1
  )Clade4:0.1,Taxon6:0.1
 )Clade3:0.1
);
```

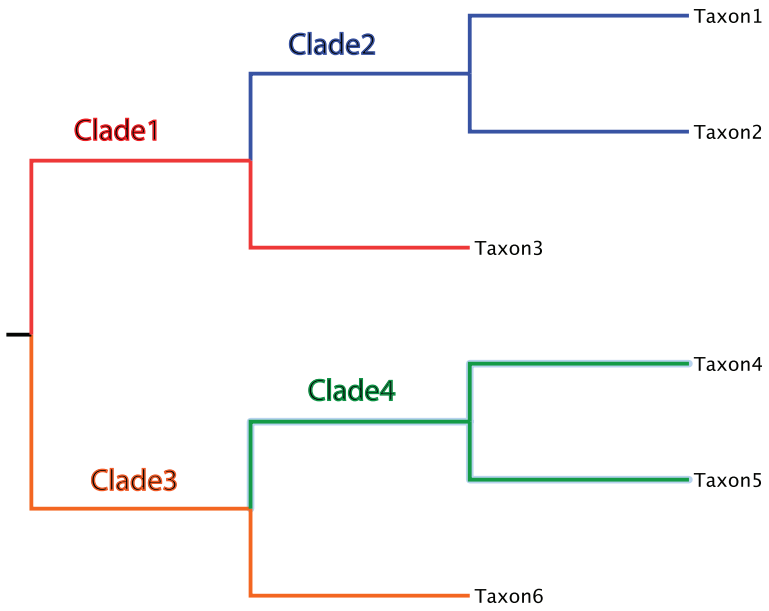


Figure 2: The subtree-labelled Newick format tree (left) and the resulting tree, which is colored to show the branches affected by the subtree label.

#### 4.3.1 Specifying Subtrees in Newick Format

To specify subtrees for lineages and motifs, the Newick Tree Format must be clade-labelled, as shown in Figure 2. For multi-partition simulation runs, note that internal node clade labels can be named arbitrarily, thus for a particular subtree, each partition can have a unique lineage-specific parameterization (e.g., Clade1_1 for partition 1, Clade1_2, for partition 2, ...). However, since taxon labels *must* be consistant between partitions, unique subtree parameters cannot be made. To specify partition-specific taxon subsequence parameters, refer to Figure 2 for the method of specifying subtrees for taxon subtrees.

| Node Type | Specification | | |
|---|---|---|---|
| Internal Node | <clade_list> | | : options; |
| Taxon | <Taxon_name> | | : options; |
| Taxon | <Taxon_name> | (partition_list) | : options; |

Figure 3: The format of subtree specification in iSGv2.

#### 4.3.2 LINEAGES

Lineages allow for a specific subtree to evolve under different conditions as specified for partitions in the tree file. iSGv2 does not allow condition changes that are not realistic, while adding options specific to lineages. Subsequence options (##) are the only set of options affected by these restrictions, shown in Table 3. Labelling ("") and indel ({}) options are no different than in partitions. *Note: In order to remove indels from a lineage, include the option {0,0} in the lineage.*

Table 3: Novel functions and restrictions imposed for lineage-specific subsequence options (##).

| Option: | Restrictions | Novel function |
|---|---|---|
| *Site rates:* | | |
| a | Partition must be either gamma, discrete gamma, or uniform rates. | Change $\alpha$ parameter for gamma site rates. |
| c | Partition must be codon rates. | Change codon position frequencies. |
| g | Partition must be discrete gamma or uniform rates. | Change the number of discrete categories. |
| r | Partition must be gamma, discrete, or uniform rates. | Remove site rates (i.e., assign uniform rates). |
| *Other:* | | |
| b | Cannot change branch scale. | None. |
| f | None. | Change character frequencies. |
| i | Only affects random input sequence (i.e., [100]) | Change percentage invariable. |
| m | Must be a matrix used for specifiec character set (nucleotides vs. amino acid), cannot take value GENERAL or GTR. | Change substitution matrix. |
| p | None. | Removes *all* constraints (e.g., invariable sites, motifs, site rates) from a lineage, effectively making all changes in the lineage entirely stochastic. |

#### 4.3.3 MOTIFS

As with lineages, motifs are specified in the subtree file, and are specified to take effect along subtrees. Since motifs are site-specific, it is required to mark the sites on the root sequence.

**Motif types:** iSGv2 allows two sub-types of motifs: PROSITE-like motifs and sequence template motifs. PROSITE-like motifs specify any length of motif that follows the PROSITE regular expression pattern. Sequence template motifs are required to cover the entire sequence, and are used to place minimum and maximum length parameters on subsequences.

**Specifying sites on root sequence:** Marking sites is done in the root sequence input file, below the input root sequence or input multiple alignment. To begin marking the sequence, place a '*' corresponding to the first position in the root sequence. This *must* be present for motifs to be correctly parsed. All other positions in the input root sequence must have a corresponding '*' or motif label (anything but '*'). Motif specification characters also must be contiguous, i.e., after specifying the positions that are in a motif, the rest of the motif specification line must be '*'s. As a regular expression, the PROSITE-like motif specification is: $*^+x^+**$, where $x$ is the character for the motif specification. For template-like motifs, the regular expression is $*[0-9]^{n}$[3], where $n$ is the length of the root sequence input. There is no limit to the number of motifs that can be specified on a root sequence. Figure 4 shows some errors that can be made in specifying motifs.

**Specifying motif constraints:** After the motif-participating sites are labelled in the root sequence, specify the conditions that apply to each site in the lineage file. The fields for specifying motifs are shown in Table 4.

Note that when marking motifs, the marked positions must be the same size as the PATTERN specified. In the case of variable positions, iSGv2 requires that the maximum size of the motif (i.e., $n$ in $\mathtt{x}(m,n)$) are

[3]For multiple sequence alignment input, the regular expression is $*[[0-9],.]^n$, see Section 5.2 for details.

| Motif specification | Error | Fix |
|---|---|---|
| `aaaaa**********` | Motif specification must begin with '*' | |
| `*aaaaaa******aaaaa**` | Motif specification must be contiguous | split motif into 2 motifs, specifying each area, or make entire motif contiguous. |
| `*******aaaaaabbbbb**` | Can only use one motif marker per motif specification | split motif into 2 motifs, specifying each area, or make entire motif contiguous. |

Figure 4: Erroneous motif specifications for a root sequence of length 20, with the assumption that the motif is labelled by 'a'.

Table 4: List of the options for specifying motifs. See Figure 5 for an example of a motif specification.

| Field | Description | Motif type |
|---|---|---|
| MARKER | A single character, value can be anything except for the character '*' (see Section 5 for specifying sites for motif). This is the ID of the motif as specified in the root sequence input. | Prosite-like, template (0-9 only). |
| NAME | Not required. This is the description of the motif. | Prosite-like, template |
| PATTERN | Regular expression patterns, as below. | |
| `[list]` | A list of acceptable amino acids or nucleotides for a site, site can take value of `list`. | Prosite-like |
| {`list`} | A list of unacceptable amino acids or nucleotides for a site, site can take any value *not* in `list`. | Prosite-like |
| `char` | Invariable site, takes value of `char`. | Prosite-like |
| x | Site can take any amino acid or nucleotide value. | Prosite-like, template |
| x($l$) | Next $n$ sites can take any amino acid or nucleotide value. | Prosite-like, template |
| x($min, max$) | Variable length motif sequence. Cannot start a motif with a variable length motif position. This specifies that length of the motif positions can vary between $m$ to $n$ sites (variance caused by indels). Variable sites cannot overlap between different motifs. | Prosite-like, template |

In root sequence specification file

```
50
00000000000000000000000000000000000000000000000000
ASPISTIQAATVPDSS--EVAGKWYIVALASNTSFLREKGKMKMVMARIL
**********bbbbbbbbbbbbbb**********************
```

In motif specification file

<subtree$_i$ >:

```
    MARKER=b;
    NAME=PS00213: Lipocalin signature, ALL;
    PATTERN=[DENG]-{A}-[DENQGSTARK]-x(0,2)-[DENQARK]-[LIVFY]-{CP}-G-{C}-W-[FYWLRH]-{A}-[LIVMTA];
```

Figure 5: An example of the specifying the lipocalin motif in the single root sequence input. The sites corresponding to `x(0,2)` in this instance is specified as 0 sites, with the '-' placeholder being used to fill the maximum possible number of sites (2) that could occupy the "variable-length site".

represented in the root sequence. To make this possible, you may need to place '-' in the root sequence or root MSA, as shown in Figure 5. If a character is placed in the variable motif instead of '-', iSGv2 will assume that this position is part of the variable motif in the root sequence.

# 5 Root Sequence Input

The root sequence is specified by the square brackets (`[ ]`) in the tree file.

There are three options for root sequence input:

1. Specifying the length of root sequence. indel-Seq-Gen will randomly generate a root sequence of the specified length. For example `[40]` calls for a root sequence of length 40 characterss.
2. Root sequence input.
3. Multiple alignment input.

Make sure that the input file is in the same directory when using a root sequence input file (options 2 and 3).

## 5.1 Root Sequence Input

This option specifies that the user has a root sequence in a file for a partition.

In the tree file, the root sequence input is specified as:

```
[:<rootseq_file>]
```

The format of the file `<rootseq_file>` is:

```
<length of sequence>
<invariable array>
<sequence>
<motif_spec>
```

Figure 6 gives an example of a root sequence for conserving a Thioredoxin-fold protein sequence motif.

```
20
00003223100000000000
LARDCVLCSTWVTIALACLK
```

```
20
00000000100000000000
LARDCVLCSTWVTIALACLK
****aaaa************
```

In motif specification file

```
root:
    MARKER=a;
    NAME=Trx-like;
    PATTERN=C-[GATPLVE]-[PHYWSTA]-C;
```

Figure 6: Thioredoxin-fold proteins have a characteristic "CXXC" motif that is conserved for all proteins in the family. (Left): This root sequence input requires the CXXC motif to remain constant for all sequences created through the use of the invariable array (described later), listed above the root sequence. The serine in position 9 will also be held invariable, though insertions are allowed to occur between itself and the previous cysteine. Finally, the length of the root sequence is given by the first line. Note that this sequence is not truly a Trx-fold sequence, but an example to show the usage of the invariable array. (Right): Motif specification of the Thioredoxin active site. Motif specifications preserve both the length dependence and character subsets. Note also that the invariable array is no longer used to preserve the motif; it is recommended that when using motif specification, all non-motif positions in invariable array should be set to 0.

## 5.2 Multiple Alignment Input

This option specifies that the user has a multiple alignment of their sequence set, and wants a root sequence created from the multiple alignment.

In the tree file, the root sequence input is specified as:

`[:<ma_file>(1,2,3)]`

Where options 1, 2, and 3 stand for:

1. The range of the multiple alignment to use, where the format is *beginning*:*end*. An input of `21:67` specifies the range from the 21st to the 67th spot (inclusive) of the input multiple alignment. Default for this option is the entire range of the multiple alignment.

2. The number of sequences to choose from the multiple alignment. indel-Seq-Gen randomly with replacement selects the specified number of sequences from the multiple alignment. Default for this option is to use all sequences.

3. Method for collapsing the multiple alignment into a root sequence, either random `'r'` or consensus `'c'`. Consensus is a majority-rule method, using a coin flip to break ties. Random uses a weighted coin toss based on the character composition at the site to choose the representative character, except for invariable positions, which will be chosen by consensus. For an example of the weighted coin toss, look at the first column in Figure 7 in which all sequences emit an amino acid, column 6. In this column, there are 2 T's, 1 V, and 1 C. A weighted coin toss on this column will be a T 50% of the time, a V 25% of the time, and a C 25% of the time. The default for this option is consensus.

When specifying the multiple alignment in the tree file, a blank field specifies that the default entry for the field is desired. For example, `[:input_MA(,,r)]` indicates that the entire range and all sequences from `input_MA` will be used, but that the character that will represent the column will be chosen by a weighted coin toss based on the characters that appear in that column. Note that the size of the root sequence can fluctuate between simulation runs. For example, using the option `[:input_MA(,1,)]` will randomly choose

Input MA, using invariable array

```
       00000000000000000000000000000000100322300010000
Taxon1 -----TKEDF-----TSDQTNPIGQDSATRLILKGEELTCNYKLFD
Taxon2 RNSCNVVPKIVQVNGDFRIRFTALRD-----IKAGEELFFNYGENF
Taxon3 -----TGVSNNQFGG---YDFVALGD-----IEVGEELTWBYETTE
Taxon4 -----CELVQLTEFS---LGVVAICN-----IEAGEELSFDYAWEA
```

Input MA, using motif specification

```
       00000000000000000000000000000000100000000010000
Taxon1 -----TKEDF-----TSDQTNPIGQDSATRLILKGEELTCNYKLFD
Taxon2 RNSCNVVPKIVQVNGDFRIRFTALRD-----IKAGEELFFNYGENF
Taxon3 -----TGVSNNQFGG---YDFVALGD-----IEVGEELTWBYETTE
Taxon4 -----CELVQLTEFS---LGVVAICN-----IEAGEELSFDYAWEA
       **********************************bbbb********
       *....00000000000000000000000..000000000000000
```

In motif specification

```
<subtree>:
    MARKER=b;
    PATTERN=G-x(2)-L;
<subtree′ >:
    MARKER=0;
    NAME=Multiple alignment template example;
    PATTERN=x(10,22)-x(2,6)-x(5)-x(3,8);
```

Figure 7: **input_MA**: (Top) An example of a 4 sequence multiple alignment from the SET-domain family of sequences using the invariable array. The GXXL motif is conserved in both the invariable array, along with the tyrosine and isoleucine. (middle, bottom) The motif specification input and root sequence file. The motif will be conserved in the same manner as the invariable array on the top. However, specifying this subsequence as a motif allows the user to conserve the motif on a subtree, rather than over the entire guide tree as is done with the invariable array.

a single sequence to create root sequence, thus the sequence that indel-Seq-Gen chooses will be the length of the root sequence for that subsequence.

Format:

```
<invariable array>
<sequence 1>
<sequence 2>
 .      .
 .      .
 .      .
<sequence n>
<motifs>
```

Figure 7 is an example of an input multiple alignment of the SET-C region of the SET-domain family. Note that the '-' is the only character that will be accepted for the gap character. Irregular characters ("YRUN" in nucleotides, "BZJX" in amino acids) cause each character for which they stand for to be counted as the 1 over the number of characters they represent (e.g., 'B' increments both 'N' and 'D' by 0.5).

Table 5: Representation of positions in the invariable array, and effects on the sequence. Invariable sites (1 and 3) block both deletion and substitution of the corresponding position in the sequence array. No-indel sites (2 and 3) block deletion of the sequence array, but also block insertion events from occurring between consecutive no-indel positions (between 2-2, 2-3, 3-2, and 3-3) in the invariable array.

| # | Effects |
|---|---|
| 0 | None |
| 1 | Invariable |
| 2 | No-indel |
| 3 | Invariable + No-indel |

```
Accepting Positions:

  Invariable Array:       0 0 1 2 3 0
  Insertion (any size):  1 1 1 1 0 1 1

  Invariable Array:       0 0 1 2 3 0
  Deletion (size 1):      1 1 0 0 0 1
  Deletion (size 2):      1 0 0 0 0 0
  Deletion (size 3):      0 0 0 0 0 0
```

Figure 8: The invariable array and accepting positions for insertion and deletion events. For insertions, the accepting positions (denoted by '1' and '0' above) are located in between consecutive positions in the invariable array, while deletion accepting positions correspond exactly those in the invariable array. In the accepting positions, the site with '1' is allowed to have indels. In the rare case that an accepting position cannot be found (as in the size 3 deletion example above), indel-Seq-Gen will output an error, but continue the simulation run.

#### 5.2.1 Multiple alignment root sequence input for template motifs

Note that the multiple alignment root sequence input will have variable length, depending on the number of columns in the multiple alignment that are inferred to be gap columns ($> 50\%$ '-' in a column). When specifying template motifs, all sites, except the first, must belong to a position in the template. For this reason, when specifying a template motif, iSGv2 also allows the character '.' on a position in the motif specification. The '.' character specifies that this site should be excluded from the template sites. For example, in Figure 7, 6 positions of the multiple sequence alignment are excluded from the template specification.

### 5.3 Invariable Array

The invariable array in indel-Seq-Gen is a quaternary array that specifies how a region is allowed to evolve, and is specified in the root sequence input. Table 5 shows the effect of each numerical entry in the invariable array, and Figure 8 shows an example of how the algorithm finds possible positions for indels based on the invariable array.

# 6 Other Input Files

## 6.1 Character Frequencies #f*freq_file*#

Protein and nucleotide subsequences often evolve under different functional constraints, causing them to display different character frequencies. For this, a file containing 20 amino acid frequencies (order: ARNDCQEGHILKMFPSTWYV) or 4 nucleotide frequencies (order: ACGT) separated by commas can be entered for each subsequence with the option: `#f<freq_file>#`. Values from the input file are read and normalized to create a distribution, where the number of values is specified by the maximum indel size of the subsequence. For an example, see the file `aaf.freq` included with all iSG `.tar.gz` archive downloads.

## 6.2 Indel Probabilities {#$_1$,#$_2$,*indel_file(s)*}

Insertion and deletion frequencies can be provided for each subsequence. The format of the file is shown in Figure 9. This example is for the indel probabilities of sizes 1–10 of the Zipfian distribution described in Chang and Benner [1]. indel-Seq-Gen will read in the number of values corresponding to the `max_indel_size` (#$_1$) specified for the subsequence (for Figure 9, the maximum indel size of a subsequence using this file can be up to size 10), and then normalize the values to create a distribution. This means that the frequencies can be given in absolute numbers or in any scale (as shown in Figure 9). If the maximum indel size is greater than the number of indel positions in the length distribution file, indel-Seq-Gen will output a message that it is unable to read the input distribution file.

```
2628,743.8,355.5,210.5,140.2,100.6,76,59.6,48.1,39.7
```

Figure 9: An example length distribution input file. This has the frequencies of indels with lengths from 1 to 10 amino acids, taken from the first 10 values of the Zipfian distribution. The number of values in this file should not be smaller than the given max indel size.

# 7 Examples

## 7.1 Basic coding and non-coding

Despite the complexity that exists in iSGv2, the main goal of the simulation method is to be an all-purpose sequence simulator. Therefore, included in the distribution is a file called "`simple_nuc.tree`", shown in Figure 10.

Given this example, assume the usage of the Hasegawa, Kishino, and Yano [2] substitution matrix. The following two commands will create non-coding and coding simulation runs, respectively:

```
./indel-seq-gen --matrix HKY --outfile DNA_out < simple_nuc.tree
./indel-seq-gen --matrix HKY --codon_rates 0.2,0.05,0.75 --outfile DNA_out < simple_nuc.tree
```

This will also create five outfiles: DNA_out.ma, DNA_out.seq, DNA_out.root, DNA_out.trace, and DNA_out.verb. In the first examples, the sites are mutated uniformly as is non-coding DNA. In the second example, the third coding mutates 75% of the time, while the first and second codn mutate 20% and 5% of the time, respectively, mimicking codon position rates.

```
[999]((Taxon1:0.3,Taxon2:0.14):0.5,(Taxon3:0.34, Taxon4:0.5):0.12);
```

Figure 10: A simple example of a nucleotide simulation partition file. This simulates a 999 nucleotide sequence with no indels, along a guide tree of 4 taxa.

mid_nuc_coding.tree

```
[999]{9,0.1/0.3,codonLD}((Taxon1:0.3,Taxon2:0.14):0.5,(Taxon3:0.34, Taxon4:0.5):0.12);
```

codonLD

```
0,0,3222,0,0,233, 0,0, 0.23
```

mid_nuc_noncoding.tree

```
[999]{9,0.1/0.3,idLD}((Taxon1:0.3,Taxon2:0.14):0.5,(Taxon3:0.34, Taxon4:0.5):0.12);
```

idLD

```
2628,743.8,355.5,210.5,140.2,100.6,76,59.6,48.1,39.7
```

Figure 11: The specification files for a simulation run with indels, and the associated files, codonLD and idLD.

## 7.2 Including indels

A little more complex example including indels is shown in Figure 12. Notice that we now need to two trees to represent coding or non-coding sequences, since indel sizes that are not a multiple of three would cause a nonsense mutation.

The commands:

```
./indel-seq-gen -m HKY -e mid_noncoding --num_runs 5 < mid_nuc_noncoding.tree
./indel-seq-gen -m HKY -e mid_noncoding -n 5 -c 2,1,8 --invar 0.02 < mid_nuc_coding.tree
```

In this example, the previously introduced options `--matrix`, `--outfile`, and `--codon_rates` have been replaced by their short options. The `--num_runs (-n)` option has also been included, so that five runs will be simulated. The multiple alignments of this example is very gappy, since in the indel options, the maximum indel size is 9 nucleotides, and an insertion occurs once for every 10 substitutions, a deletion once every 3.3 substitutions. Finally, the option `--invar` sets 2 percent of the sites to be invariable. Codon rate input also does not need to add up to 1, as iSGv2 will normalize the values.

## 7.3 Coding and non-coding together

iSGv2 has the unique ability to simulate exons and introns in a single run. This adds yet more complexity, as you can see by the necessary files, as listed in Figure 11. Files that remain the same as previous examples are excluded from this figure.

The command:

```
./indel-seq-gen -m HKY -c 2,1,8 -e exon_intron --step_type trs < exon_intron.tree
```

There are many interesting things to note:

1. Codon rates are input at the command line. The subsequence option #r# in the intron sequence resets the rates to uniform rates in the intron. The additional subsequence options #b1.2# multiplies all branch lengths in the tree by a factor of 1.2.

2. The file `intron` conserves the nuclear-like spliceosomal sites (GU at beginning, G in the middle, and AG at the end).

exon_intron.tree

```
[:exon(,,)]{9,0.031/0.01,codonLD}((Taxon1:0.3,Taxon2:0.14):0.5,(Taxon3:0.34, Taxon4:0.5):0.12);
[:intron]#r,b1.2#{9,0.1,idLD}((Taxon1:0.3,Taxon2:0.14):0.5,(Taxon3:0.34, Taxon4:0.5):0.12);
```

exon

```
          000000000000000000000000000000000000000000
Taxon1    TTACTTT---TTCCTAACCGG---C---CCGAGCT---AATG
Taxon2    TTAGTTTTTAATCCCAACCTG---C---CCCATCT---GATA
Taxon3    CGAAATC------TTACCGGAGGTAATACGCGGCGC---ATC
Taxon4    CTGCATT------CAACCTGT---T---AGCATCGC---ATC
```

intron

```
30
330000000000001000000000000033
GUTTCAGGTAAAATGCANNGACTTAGRYAG
```

Figure 12: The specification files for a simulation run of one exon partition and one intron partition with indels, and the associated root sequence files, exon and intron.

3. The option `--step_type` sets the simulation run to be in time relative steps. The output file `exon_intron.trace` will contain where each indel event occurred and the relative time of the event, as explained in Section 4.1.1.

## 7.4 Lineage Introduction

Lineages are one of the unique abilities of iSGv2, paricularly the ability to cause a lineage to evolve under no specifications, which we call "pseudogene" evolution (the `p` option in subsequence options `##`). To add lineages to the previous example, we need to specify clades in the partition file, create the lineage file and specify this file on the command line. Figure 13 shows the file changes, and Figure 14 shows the output of the command:

```
./indel-seq-gen -m HKY --lineage exon_intron_lineage.spec --alpha 1.3 < exon_intron_lineage.tree
```

No output files are given, thus the output is printed to the screen. Note that the file `outfile.verb` will still be created.

Some finer points:

1. The global parameter `--alpha` specifies that, unless otherwise specified in clades, all branches in all trees will evolve with site rates, where the gamma parameter $\alpha = 1.3$. However, Clade1_1 and Clade1_2 both become pseudogene lineages, which releases site rates from them. Taxon3 in the first tree also does not evolve with this gamma, since the lineage file specifies it to evolve with $\alpha = 0.7$ (`#a 0.7#`).
2. Taxon3, partition 1, evolves without indels ({0,0}).

## 7.5 Prosite-like motif

To create Prosite-like motifs, a root sequence input is necessary. In this example, the lipocalin root sequence and motif specification in Figure 5 is used, along with `lipocalin.tree`, shown in Figure 7.5.

```
indel-seq-gen -m JTT -k lipocalin.spec -w a < lipocalin.tree
```

Figure 15 shows the output of this command.

exon_intron_lineage.tree

```
[:exon(,,)]{9,0.031/0.01,codonLD}
(
 (Taxon1:0.3,Taxon2:0.14
 )Clade1_1:0.5,
 (Taxon3:0.34, Taxon4:0.5
 ):0.12
);
[:intron]#r,b0.5#{9,0.1,idLD}
(
 (Taxon1:0.3,Taxon2:0.14
 )Clade1_2:0.5,
 (Taxon3:0.34, Taxon4:0.5
 ):0.12
);
```

exon_intron_lineage.spec

```
LINEAGES =
{
    Clade1_1:#fpse.freq,p#{5,0.08,idLD};
    Clade1_2:#p#{8,0.08,idLD};
    Taxon3(1):#a 0.7#{0,0};
}
```

pse.freq

```
10,23,15,6
```

Figure 13: The specification of subtrees in exon_intron_lineage.tree and the corresponding specifications in exon_intron_lineage.spec. Subtrees on internal nodes can be uniquely named, thus the convention Clade_<partition_number>. Taxa cannot be uniquely specified between partitions, since they are required to be the same for all partitions. Thus, the taxon lineage (Taxon3) contains the list of partitions it is active in in parentheses. In this case, the lineage specifications for Taxon3 are active only for the first partition.

```
>Dataset_0__partition_1
[0,D,,1100,4,16:17:18:19]
[1,I,,1100,1,23]
[2,D,,1100,4,30:31:32:33]
[3,D,,1000,1,15]
[4,I,,1000,2,21:22]
[5,I,,0011,3,12:13:14]
>Dataset_0__partition_2
[6,D,,1000,2,59:60]
[7,I,,1000,4,39:40:41:42]
[8,D,,1000,6,61:62:63:64:65:66]
[9,D,,0001,1,62]
CTACATTATCCTAACCGGCCCCAGCGCAATCGTTTCAGGTAAAATGCACAGACTTAGACAG
 4 61
Taxon1    CTACAGCAGCCACCGACCACCCGCCGCCGTTGTCAGTTGGTCTGCAGCCAG
Taxon2    CTACAGCAGCCTAGGCAAGAACCCGTGTCAGTTGAACTGCAAAGACTCAGCCAG
Taxon3    ATACTGTCTCCCAATACCCAGCGCTATAGCAAATGTATAAGGTAAAATGGACAGACTTTGACAG
Taxon4    TTATCTTAGGACAGTACCCAGACCCCGGGCAATCGTGTGAACTAGAACGCAGAGCATAGCCAG
 4 61
Taxon1    CTACAGCAGCC--------ACCGACCACC----CGCCGCCGTTGTCAGTTGGTCTGCA--
Taxon2    CTACAGCAGCC---T----A--GGCAAGA----ACCCG----TGTCAGTTGAACTGCAAA
Taxon3    ATACTGTCTCCCAATACCCA---GCGCTATAGCAAATG----TATAAGGTAAAATGGACA
Taxon4    TTATCTTAGGACAGTACCCA---GACCCCGGGCAATCG----TGTGAACTAGAACGCAGA

Taxon1    ------GCCAG
Taxon2    GACTCAGCCAG
Taxon3    GACTTTGACAG
Taxon4    G-CATAGCCAG
```

Figure 14: The output for a lineage simulation, consisting of event tracing, root sequence, sequences, and multiple alignment. Taxon1 and Taxon2 are "pseudogenes".

lipocalin.tree

```
[:lipocalin]
{5,0.1,idLD}
(((Taxon1:0.4,Taxon2:0.14):0.5,Taxon3:0.34):0.1, Taxon4:0.5);
```

```
>Dataset_0__partition_1
[0,D,,1100,1,10]
[1,D,,1100,2,9:11]
[2,I,,1000,5,44:45:46:47:48]
[3,D,,1000,5,50:51:52:53:54]
[4,D,,1000,2,8:12]
[5,D,,1000,2,35:36]
[6,D,,1000,1,42]
[7,D,,1000,2,39:40]
[8,I,,0100,2,18:19]
[9,D,,0001,1,42]
[10,D,,0001,1,52]
[11,D,,0001,2,55:56]
[12,I,,0001,1,4]
ASPISTIQAATVPDSSEVAGKWYIVALASNTSFLREKGKMKMVMARIL
 7 48
5         ASPISTIQAATVPDSSEVAGKWYIVALASNTSFLREKGKMKMVMARIL
6         DSPIDTIWAAVVPDSSDIAGRWYLMALVSDTSFLREKAKLKMVVAGVL
7         DRPVDMSVVGDSSAVDGTWLFMQYVTEVSFVRQRLKFKMLVSTKL
Taxon1    EGPLDTAVEEEQISGNWLGMRSTYHVVSLFNQGILQEV
Taxon2    ERSLEMTVVGDSSGDAIDGVWLFLQYINEVGFLRQRLKFAALGTIKL
Taxon3    DGPIDTIQTNIVLDSSDIAGRWYVMDLIGDAMFRRTMKGLKNVLSGPL
Taxon4    PFPNLATKLIGVQPDQDEIIGQWYELSHHSKSAIFGDTSMKLVTK

 7 48
5         ASP-ISTIQAATVPDSS--EVAGKWYIVALASNTSFLREKGKM-----KMVMARIL
6         DSP-IDTIWAAVVPDSS--DIAGRWYLMALVSDTSFLREKAKL-----KMVVAGVL
7         DRP-VDMS---VVGDSS--AVDGTWLFMQYVTEVSFVRQRLKF-----KMLVSTKL
Taxon1    EGP-LDT-----AVEEE--QISGNWLGMRSTYHV--VS--L-FNQGILQ-----EV
Taxon2    ERS-LEMT---VVGDSSGDAIDGVWLFLQYINEVGFLRQRLKF-----AALGTIKL
Taxon3    DGP-IDTIQTNIVLDSS--DIAGRWYVMDLIGDAMFRRTMKGL-----KNVLSGPL
Taxon4    PFPNLATKLIGVQPDQD--EIIGQWYELSHHSKSAIFGDTS-M-----KLV-TK--
```

Figure 15: The output for a lipocalin simulation, consisting of event tracing, root sequence, sequences, and multiple alignment.

## 7.6 Template motif

To set up a run with a template, we will using the true multiple alignment sequences output in Figure 15 as the input root sequence. The resultant file and template specifications are given in Figure 7.6, and we use `lipocalin_ma.tree`, a tree similar in Figure 7.5, where `[:lipocalin]` is replaced by `[:lipocalin_ma(,,)]`.

`lipocalin_ma`

```
          0000000000000000000000000000000000000000000000000000000000
Taxon1    EGP-LDT-----AVEEE--QISGNWLGMRSTYHV--VS--L-FNQGILQ-----EV
Taxon2    ERS-LEMT---VVGDSSGDAIDGVWLFLQYINEVGFLRQRLKF-----AALGTIKL
Taxon3    DGP-IDTIQTNIVLDSS--DIAGRWYVMDLIGDAMFRRTMKGL-----KNVLSGPL
Taxon4    PFPNLATKLIGVQPDQD--EIIGQWYELSHHSKSAIFGDTS-M-----KLV-TK--
PS00213   **************bbbbbbbbbbbbbb**************************
lipo_ma   *77777777777777777777777777777777777777777.....77777777
```

`lipocalin_ma.spec`

```
MOTIFS =
+
{
        root:
                MARKER=b;
                NAME=PS00213: Lipocalin signature;
              PATTERN=[DENG]-{A}-[DENQGSTARK]-x(0,2)-[DENQARK]-[LIVFY]-{CP}-G-{C}-W-[FYWLRH]-{A}-[LIVMTA];
        root:
                MARKER=7;
                NAME=lipo_ma: Lipocalin partial template;
                PATTERN=x(5,20)-x(10,30);
}
```

Note that in this example, there is a very tight restriction on the number of insertions, while deletions are more acceptable. Figure 16 shows these restrictions based on the input root sequence. Note that the sites marked by '.' are removed consideration when building the root sequence.

To run this example, simply type:

```
./indel-seq-gen -m JTT -k lipocalin_ma.spec < lipocalin_ma.tree
```

Note that this example incorporates both a motif and template, and that they do not need to coincide. This is often the case for real protein evolution, such as the lipocalins in this example, where the lipocalin signature begins in a coil region and ends in a beta strand. A good use of the template for this type of protein sequences is to make each $x(min, max)$ match with the coil regions and beta strands, as is done in the published work [4].

```
template:  ---- x(5,20), 17 --- -------- x(10,30), 30 --------
positions: 12345678911111111112 12345678911111111122222222223
                     01234567890          01234567890123456789O
             *             **

Taxon1    EGP-LDT-----AVEEE--QI SGNWLGMRSTYHV--VS--L-FQ-----EV
Taxon2    ERS-LEMT---VVGDSSGDAI DGVWLFLQYINEVGFLRQRLKFAALGTIKL
Taxon3    DGP-IDTIQTNIVLDSS--DI AGRWYVMDLIGDAMFRRTMKGLKNVLSGPL
Taxon4    PFPNLATKLIGVQPDQD--EI IGQWYELSHHSKSAIFGDTS-MKLV-TK--
lipo_ma   *77777777777777777777 77777777777777777777777777777
```

Figure 16: The input multiple alignment `lipocalin_ma` from Figure 7.6 as iSGv2 reads it in. Above the alignment is the placement of the template sites as iSGv2 places them; `template:` Gives the template spec (e.g., `x(5,20)`) and the current number of sites residing in the template as iSGv2 reads the root sequence, `positions:` the position in the template. Numbers with a '*' below them are positions that are not included in the root sequence built. A space has been placed between templates to make it easier to read.

# References


[1] Mike S. S. Chang and S.A. Benner. Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. *Journal of Molecular Biology*, 341:617–631, 2004.

[2] M. Hasegawa, H. Kishino, and T. Yano. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 22:672–677, 1985.

[3] A. Rambaut and N.C. Grassly. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *CABIOS*, 13(3):235–238, 1997.

[4] C.L. Strope, K. Abel, S.D. Scott, and E.N. Moriyama. Biological sequence simulation for complex evolutionary hypotheses with indel-Seq-Gen version 2. *in prep*, x, 2008.

[5] X. Xia and Z. Xie. Protein structure, neighbor effect, and a new index of amino acid dissimilarities. *Mol. Biol. Evol.*, 19:58–67, 2002.