Instructions on using the `intronRates` package

Miklós Csűrös
e-mail:csuros AT iro.umontreal.ca

April 19, 2007

1 Introduction

The Java package `intronRates.jar` contains a number of programs and supporting Java classes for modeling and analyzing intron evolution. The package implements likelihood methods [Csűrös, Holey, Rogozin "In search of lost introns", *ISMB*, 2007] for analysis of intron data.

The programs may be useful for the phylogenetic analysis of any 0-1 data set where rates may vary on the branches.

Please send me a note if you use the package. I would send you an e-mail whenever the package is updated. (I am extending the model to include across-sites variation, and other features.)

The main change from a previous version (`intronLoss.jar`) is that likelihoods are computed and optimized much faster, and the option for keeping certain gain/loss rates constant.

2 File types

The programs work with the following main file types.

⟨**tree file**⟩ describes a phylogeny in the Newick format (`http://evolution.genetics.washington.edu/phylip/newicktree.html`) used by Phylip and other programs. It is good to keep it unrooted (with one inner node that has three descendants) because otherwise the root's position is not unique (can "slide" between its two children: it is impossible to determine the branch length to those two guys). Name the inner nodes, it helps deciphering the outputs. Tree branch lengths are not important for the optimization (since they will be optimized too).

⟨**table file**⟩ A file of aligned 0-1 sequences for intron presence-absence in homologous positions. Lines starting with `#` are skipped (supposed to be comments and such). Data lines have two fields, separated by TABs: taxon name (must match name in ⟨tree file⟩) and 0-1 string of intron presence/absence. (This is the format used by [Rogozin et al "Remarkable interkingdom conservation of intron positions and massive, lineage-specific intron loss and gain in eukaryotic evolution", *Current Biology*, **13**:1512–1517, 2003].)

⟨**rate file**⟩ An intermediate text file listing loss and gain rates along branches. This file is the output of the program `calculateConstantRates`.

3 Main programs

All programs are implemented in Java for portability. In order to run program X, you need to invoke it through the Java engine:

`java -cp intronLoss.jar ca.umontreal.iro.evolution.introns.`X ...

If you run out of memory, you may need to use the `-Xmx` switch as in `java -Xmx512M ...` The programs write to the standard output, so you need to redirect or pipe it through something to save the result. The output typically begins with some information on who, when and how produced the text, written in comment lines that start with `#|`.

3.1 Program `calculateConstantRates`

Produces a ⟨rate file⟩ by fitting the parameters (gain & loss rates, length on each branch and root's intron density) to the data to maximize likelihood.

> `calculateConstantRates` [⟨switches⟩] ⟨table file⟩ ⟨tree file⟩. Optimize parameters starting from default initial values.

The following switches are implemented. (d stands for a double-precision floating-point and i stands for an iunteger).

opt-eps Specifies a stopping rule for the optimization. The optimization stops if the log-likelihood (natural logarithm) changes by less than this value between rounds. Syntax: `-opt-eps` d.

opt-round Specifies a stopping rule for the optimization. The optimization stops after that many rounds. Syntax: `-opt-round` i.

start-with Specifies that the optimization should start with saved values from a previous execution of `calculateConstantRates`. Syntax: `-start-with` ⟨rate file⟩.

fixrate Fixes the gain or loss rates on certain branches. Syntax: `-fixrate` ⟨constraints⟩, where ⟨constraints⟩ comma-separated edgewise constraints. Edgewise constraints have the format ⟨clade⟩:⟨rate⟩ or ⟨clade⟩:⟨rate⟩:⟨rate⟩, where ⟨clade⟩ is a taxon name in your Newick file (the bottom node for the branch), and ⟨rate⟩ has the syntax Ld or Gd fixing Loss or Gain rate at d. Example: `java -cp intronRates.jar ca.umontreal.iro.evolution.introns.calculateConstantRates -fixrate 'Diptera:G0:L00.01,Ascomycota:G0' rogozin_introns_table tree.tre`.

3.2 Program `computeConstantPosteriors`

Outputs mean values for intron counts and intron gains/losses.

`computeConstantPosteriors` [⟨switches⟩] ⟨table file⟩ ⟨tree file⟩ ⟨rate file⟩
Parameters as in calculateConstantRates but there is no more optimization performed.

Currently the only implemented swicth is `-ambiguous` in the same syntax as above.

4 An example session

This session illustrates how to use the programs in conjunction with the data of Rogozin et al. (2003).

1. Download the data file `ftp://ftp.ncbi.nlm.nih.gov/pub/koonin/intron_evolution/rogozin_introns_table`. For simplicity, name it `table.txt`. The data is for 7236 homologous intron positions in 684 orthologous gene sets, across 8 species: *D. melanogaster* (Dr), *A. gambiae* (An), *C. elegans* (Ca), *H. sapiens* (Ho), *S. ecervisiae* (Sa), *S. pombe* (Sc), *A. thaliana* (Ar), and *P. falciparum* (Pl). This will be your ⟨table file⟩.

2. Choose your phylogeny. Here are two possibilities:

```
(((((Dr, An) Diptera, Ca) Ecdysozoa, Ho) Bilateria,
(Sc, Sa) Ascomycota) Opisthokont, Ar, Pl) Crown;
[ tree with ecdysozoan clade (nematoda-arthropods)]
```

   or

```
(((((Dr, An) Diptera, Ho) Coelomata, Ca) Bilateria,
(Sc, Sa) Ascomycota) Opisthokont, Ar, Pl) Crown;
[ tree with coelomate clade (vertebrates-arthropods)]
```

Save your phylogeny in a file called `tree.tre`, this will be your ⟨tree file⟩.

3. Optimize the rate parameters.

```
java -cp intronRates.jar \
   ca.umontreal.iro.evolution.introns.calculateConstantRates \
   table.txt tree.tre > rates.txt &
```

The execution should not take more than a minute or so if you have a computer built after 2003. If you want to see what is going on, use instead the `-v true` switch and pipe through `tee rates.txt`. The file `rates.txt` lists the probabilities for gain and loss for each node (on the edge that leads to it), and from the comments you can decypher the actual gain and loss rates and branch lengths — the rates and lengths are scaled on each edge that the loss and gain rates add up to 1.

4. Check the posterior counts.

```
java -cp intronRates.jar \
   ca.umontreal.iro.evolution.introns.computeConstantPosteriors \
   table.txt tree.tre rates.txt  > history.txt
```

The output gives two estimates for every quantity: one for intron sites in the data ("current") and another one that accounts for introns that were completely lost in the course of evolution ("with missing sites" or "all"). This latter is the true estimate of the unknwon quantities.