Guide to estimating the distribution of selection coefficients using the TdG12 site-wise mutation-selection model

May 23, 2012

1 Introduction

This document describes how to use the TdG12 site-wise mutation-selection model ('swMutSel0') to estimate the distribution of selection coefficients (or 'fitness effects') from an alignment of protein coding sequences.

1.1 Requirements

The steps in this tutorial have been tested on Linux and Mac OS X 10.6+. Our program does not estimate tree topology or perform branch length optimisation, therefore we recommend the use of two popular tools for this purpose: RAxML and PAML[1]. Both have Windows versions available for download, but they have not been tested by us. We assume that you already have them installed and are able to run them on your computer. Our software is written in Java and should run on all platforms that have the Java Runtime Environment (JRE) (6 or later) installed. The JRE is available for Windows, Linux and Mac OS X 10.6+.

1.2 Installation

You can download the executable files for the program from http://mathbio.nimr.mrc.ac.uk/. Download the tdg12.zip file and extract the contents. The download contains the files required for this tutorial and `tdg12.jar`, which is the Java binary file for the program.

1.3 Citation

Tamuri AU, dos Reis, M and Goldstein RA (2012) Estimating the Distribution of Selection Coefficients from Phylogenetic Data Using Sitewise Mutation-Selection Models. (December 29, 2011, doi: 10.1534/genetics.111.136432)

[1]However, you can use other programs if you prefer.

2 An example analysis

2.1 Preparing the alignment and tree

[[Basic background? Types of sequences, not too many gaps, no recombination etc. You need good alignments and enough divergence??]]

For the purposes of this tutorial, we will be estimating the distribution of selection coefficients of a set of mammalian mitochondrial ATP8 protein coding genes. The download provides an alignment, atp8.phyl (which was built using PRANK) and a tree, atp8.tree (estimated by RAxML with branch lengths optimised using PAML's codeml). Full details are available from the program's websites. The options used for running RAxML were:

```
raxmlHPC -f a -x 12345 -p 12345 -N 100 -m GTRGAMMA -s atp8.phy -n
atp8
```

and the PAML codeml control file (codeml.ctl) is available in the download. We optimised the branch lengths using the FMutSel0 model in codeml.

2.2 Getting estimates of global parameters from codeml

There are a number of global, site-invariant, parameters that are required for the TdG12 model. They are:

1. τ (tau) - rate of multiple substitutions.
2. κ (kappa) - transition/transversion bias.
3. π (pi) - base nucleotide composition.
4. μ (mu) - branch scaling.

Each of these parameters can be estimated under the TdG12 model but this requires significant computational resources and the use of a distributed version of the software (not covered here). However, you can get obtain good estimates for branch lengths, κ, π and μ from the faster PAML codeml analysis. The results file (named 'mlc') contains the new tree (with re-estimated branch lengths) and estimates of κ and π. An estimate of μ can be calculated from $\mu = 3 \times T_{dS}/T$, where T_{dS} is the tree length in dS (synonymous changes) units and T is the total tree length, also found in the codeml 'mlc' results file. *Remember to save the tree from the 'mlc' file and use that for the TdG12 program.*

Therefore, the estimates for the global parameters for the ATP8 gene alignment are:

$\kappa = 4.93022$

$\pi = \{0.25299, 0.22268, 0.43860, 0.08572\}$

$\mu = 3 \times T_{dS}/T = 3 \times 70.7207/113.24814 = 1.8734.$

We choose a small number for τ, e.g. 1.0×10^{-2}.

2.3 Analysing the data using the site-wise mutation-selection model

You can now run the TdG12 program to estimate the site-wise fitness of each amino acid in our alignment. The available options for running the program are:

-t The tree file in NEWICK format (required). Remember to use the tree supplied by PAML's codeml.

-s The protein coding alignment file in PHYLIP sequential format (required).

-gc The genetic code, 'standard' or 'vertebrate_mit' (required).

-tau Rate of multiple substitutions (required).

-kappa Transition/transversion bias (required).

-pi Comma-separated (with no spaces) base nucleotide frequencies (T,C,A,G) (required).

-mu Branch scaling factor (required).

-useapprox Use the approximation for unobserved residues which allows for faster computation (optional).

-site Location of the site to analyse (optional, default = all sites in the alignment).

-optimruns The number of restarts for the optimisation routine (optional, default = 1).

-threads The number of threads to use for processing in a multicore environment (optional, default = 1).

You start the analysis of the set of ATP8 genes by running:

```
java -cp tdg12.jar tdg.Analyse -s atp8.phy -t atp8.tree -gc vertebrate_mit
-tau 1e-2 -kappa 4.93 -pi 0.2530,0.2227,0.4386,0.0857 -mu 1.8734 >
tdg.out
```

You must add '`> tdg.out`' to redirect the analysis output to a file named tdg.out.

Using multicore/multiple CPUs

If you are running the program on a computer with multicore or multiple CPUs, you can specify the **-threads** option. Usually, this would be the number of available cores − 1. For example, to utilise 3 cores you would run:

```
java -cp tdg12.jar tdg.Analyse -s atp8.phy -t atp8.tree -gc vertebrate_mit
-tau 1e-2 -kappa 7.8 -pi 0.25,0.25,0.25,0.25 -mu 2.3 -threads 3 >
tdg.out
```

2.4 Parsing the results and calculating the distribution

Once the program completes, the results saved in tdg.out need to be processed to calculate the distribution of selection coefficients. The output looks something like (truncated):

```
Site 1 - Residues:  [1/20] { 1:(12, M), 2:(0, A), 3:(1, R), ...  }
Site 1 - Optimisation run (267 evaluations).  lnL = -4.939901E-5, Params = {0.0, ...}
Site 1 - Homogeneous model lnL: -4.939901011180276E-5
Site 1 - Fitness:  { -13.06494, -18.20550, -16.32137, ...  }
Site 1 - Pi:  { 4.79931E-6, 5.79345E-8, 2.70189E-7, ...  }
...
```

The output is quite verbose. Run the following command (specifying the name of the result file with '-o tdg.out'):

```
java -cp tdg12.jar tdg.results.All -o tdg.out -gc vertebrate_mit -tau
1e-6 -kappa 7.8 -pi 0.25,0.25,0.25,0.25 -mu 2.3
```

The global parameters should be the same as were specified when you run the analysis. This command reads the results file and the global parameters from the command-line, and writes the files:

F.txt Fitness values for each amino acid at each site

Q0.txt Neutral mutation matrix for entire alignment

S.txt Selection coefficients matrix for each site

QS.txt Mutation with selection matrix for each site

PiS.txt Codon frequencies at each site

PiAA.txt Amino acid frequencies at each site

distribution.mutations.csv the distribution of selection coefficients for mutations

distribution.substitutions.csv the distribution for substitutions.

The last two are comma-separated value files that can be opened in a program like Excel to plot the distribution. The three columns in the distribution files are (i) the histogram bins for S (ii) all mutations or subsitutions and (iii) non-synonymous mutations or substitutions.

3 Simulating data using the mutation-selection model

3.1 Simulating a single set of fitnesses (for one site)

```
java -cp tdg12.jar tdg.results.All -tree sim.tree -output
out.phy -sites 100 -fitness 0.0,0.2,0.3,1.0,0.5,0.6,0.7,0.8
-characters A,R,N,D,C,Q,E,H -tau 0 -pi 0.25,0.25,0.25,0.25 -mu
1.0 -gc standard
```

This command will write an alignment file 'out.phy' (in PHYLIP format) simulating a site (100 times) on the tree 'sim.tree' with the specified fitnesses for the specified characters. Characters that do not appear in the -characters option are not observed at that site (i.e. have fitness of $-\infty$). Set the global parameters as required.

3.2 Simulating multiple sets of fitnesses (for multiple sites)

Create a file with 20 fitnesses values on each line, each line corresponding to a single location in the alignment. Each fitness must be separated by a space and each site should be on a new line. For example, to simulate an alignment with 5 sites:

```
-21 -21 -21 -21 -21 -21 -21 -21 -21 2.19 3.07 -21 2.39 -21 -21 -21 -21 -21 -21 -21
-21 -21 1.74 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 4.23 -21 -21 -21 -21
0.93 -21 -21 -21 -21 -21 -21 -21 -21 0.31 -21 -21 2.63 -21 -21 -21 4.11 -21 -21 6.39
2.75 -21 -21 -21 -21 -21 -21 -21 -21 0.92 -21 -21 1.71 -21 2.28 0.72 1.33 -21 -21 3.51
-21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 -21 0.79 -21 -21 -21 -21
```

Each line specifies the fitnesses of each amino acid in the canonical IUPAC order. Residues that are unobserved at a site are given a fitness < -20. If this file is saved as 'F.txt', the alignment is generated using the command:

```
java -cp tdg12.jar tdg.results.All -tree sim.tree -output
out.phy -fitnessfile F.txt -tau 0 -pi 0.25,0.25,0.25,0.25 -mu
1.0 -gc standard
```

4 Colophon

TdG12 uses the following libraries:

1. Colt Project (http://acs.lbl.gov/software/colt/). For linear algebra.
2. PAL: Phylogenetic Analysis Library (http://www.cebl.auckland.ac.nz/pal-project/). Reading/traversing/writing NEWICK trees and PHYLIP alignments.
3. Apache Commons Math (http://commons.apache.org/math/). For optimisation.
4. Guava: Google Core Libraries (http://code.google.com/p/guava-libraries/).
5. JCommander (http://jcommander.org/). Parsing and managing command-line options.
6. Simple Java HTTP server (http://www.simpleframework.org/). Used by tdg.distributed.Slave.
7. Asynchronous Http Client library for Java (https://github.com/sonatype/async-http-client). Used by tdg.distributed.Master to make asynchronous HTTP requests to the slaves.