oncSpectrum:

A Likelihood Program to Fit Codon Models to Mutation Spectrum Data

Version 1.2

Introduction

The ANSI C program oncSpectrum implements the maximum likelihood models of Yang, Ro and Rannala (2003) for analyzing the p53 mutation database. Consult the paper for details of the models. Since version 1.1, I have replaced the original database file somaticR5.txt (version 5) with TP53SomaticR10.txt (version 10).

Getting started

Print out this document, the file refseq.txt, and have a look at the first few lines of the database file. Do not print as it is huge. If you use Windows, you can type oncSpectrum to run the program. Otherwise you will have to compile the program first.

Files Included

oncSpectrum.c, paml.h, tools.c: are the source files.
oncSpectrum.exe: Windows executable.
somaticR5.txt: the p53 version 5 database analyzed in Yang et al. (2003). Look at the file for the file format. This file was saved out by MS Excel, with "Tab"s delimiting fields. You can also read in this file from MS Excel. Note that the first line has variable names and is ignored by oncSpectrum. Note added in Dec 2011: somaticR5.txt is not included in the release. Instead find the file TP53SomaticR10.txt (version 10).
refseq.txt: has the reference sequence, used in Yang et al. (2003).

Compiling and Running the Program

You may have to edit the source file oncSpectrum.c and recompile the program. Try one of the following commands. The program is in ANSI C, and any compatible compiler should work. Note that the switch –o specifies the name of the output executable file, -O2 optimizes the code, and –lm links to the math library. You might want to or have to change some of those compiler flags. For example you might use gcc instead of cc, or -fast or -O4 instead of -O2, and you might not need the –lm flag.

MS Visual C++:
```
cl -O2 oncSpectrum.c tools.c
```
UNIX compilers:
```
cc -O2 -o oncSpectrum oncSpectrum.c tools.c –lm
cc -O2 -o oncSpectrum oncSpectrum.c tools.c
gcc -O2 -o oncSpectrum oncSpectrum.c tools.c –lm
```

To run the program, type one of the following.
```
oncSpectrum
oncSpectrum <datafilename> <refseqfilename>
```

The default is to read the p53 files (somaticR5.txt and refseq.txt).

Fitting Models

This program has evolved somewhat. The models implemented in the current version overlap with the models described in the paper but are not exactly the same. The program first reads the reference sequence file, which may include domain partition information as well, and then the mutation database. It will then go through the models automatically. For each model fitted, you will see output like the following

```
*** Model 10111
             silent     kappa   NonMiss       CpG       diY
           variable        No  variable       Yes       Yes
npara             5         1        12         1         1

np = 20
```

The model is set up using five flags, or answers to the following five questions.

(1) Is silent rate variable for different domains?
(2) Is kappa variable for different domains?
(3) Are nonsense and missense rates variable for different domains?
(4) Do we have a CpG rate parameter (rate from CpG over rate to CpG)?
(5) Do we have a diY (dipyrimidine) rate parameter (rate to diY over rate from diY)?

The first 3 questions do not apply if there is one domain (partition). All the combinations of the flags are evaluated by the program, so that 4 models are automatically fitted for data of one partition, and 24 models are fitted for data of multiple partitions. Note that fitting parameter-rich models requires a lot of data. If your datasets is not huge, please ignore the results for some of the complex models.

The example above is for an analysis of the p53 dataset with 6 partitions (domains). The mode 10111 involves 5 silent-rate parameters, 1 κ parameter, 6 λ_N and 6 λ_M parameters, plus two other parameters (λ_{CG}, λ_{PY}). Parameter estimates and SEs are printed out in this order. The SEs are calculated using the Hessian matrix, which is calculated using the difference approximation and which may not be reliable; -1 indicates an error. Parameters are bounded during the iteration (the bounds may be 0.001, 999 but I have noted that those bounds tend to be too wide, causing numerical problems). Run the same analysis 2 or 3 times to confirm that the results are stable between runs. For the parameter-rich models, the iteration may be easily stuck at the boundary of the parameter space, so run the program a few times and use the set of results with the highest lnL.

Table 1. Some of the models implemented in the program

Model	*p*	Parameters	Fig. 1	lnL
00000	3	κ, α_N, α_M	M0	–140,823.93
00001	4	κ, α_N, α_M, λ_{PY}		
00010	4	κ, α_N, α_M, λ_{CG}	M3	
00011	5	κ, α_N, α_M, λ_{CG}, λ_{PY}	M5	
00100	$2n + 1$	κ, Λ_N, Λ_M	M1	–131,201.71
01100	$3n$	$\underline{\kappa}$, Λ_N, Λ_M	M2	–130,967.14
00110	$2n + 2$	κ, Λ_N, Λ_M, λ_{CG}	M4	
01110	$3n + 1$	$\underline{\kappa}$, Λ_N, Λ_M, λ_{CG}	M5	
01111	$3n + 2$	$\underline{\kappa}$, Λ_N, Λ_M, λ_{CG}, λ_{PY}	M7	–126,000.21 (v1.1)
				–126,038.53 (v1.2)
11111	$4n + 1$	Λ_S, $\underline{\kappa}$, Λ_N, Λ_M, λ_{CG}, λ_{PY}	M6	–125,835.26 (v1.1)
				–125,871.75 (v1.2)

Note.— *n* is the number of partitions (domains).
$\underline{\kappa}$ means *n* transition/transversion rate ratios κ for *n* partitions;
Λ_N means *n* nonsense rates α_N for *n* partitions;
Λ_M means *n* missense rates α_M for *n* partitions.
λ_{CG}: mutation rate from (to) CpG is multiplied (divided) by λ_{CG}.
λ_{PY}: mutation rate from (to) dipyrimidines is multiplied (divided) by λ_{PY}.
lnL is from the analysis of the p53 database V10 (TP53SomaticR10.txt).
Model M8 of fig. 1 means that each tissue type has its own set of parameters; this is achieved by running model M7 on data of each tissue type separately.
V1.1 uses quintets to deal with CpG and di-pyrimidine rates while V1.2 uses triplets (see version history for more notes).

Technical Notes

Here are some notes for two routines in the source file oncSpectrum.c.

(A) First the main() routine sets up the model, gets memory, reads in the mutation data, and then calls the optimization routine ming2() to minimize the negative log likelihood, -lnL. Random numbers are used as initial values for the iteration. When the program runs, it prints out many lines like the following on the screen during the iteration. A healthy iteration is indicated by p going to 0, and –lnL stabilizing.

```
 8 h-m-p  1.6000 8.0000   0.0006 Y    99341.856572  0 1.1986   97 | 0/3
 9 h-m-p  1.6000 8.0000   0.0000 Y    99341.856572  0 1.6000  106 | 0/3
```

You should run the same analysis a few times to make sure that the algorithm converges to the MLEs. The optimizer or the file tools.c is from the paml package, and I did not bother to remove the irrelevant routines in it.

(B) The routine ReadMutData() reads in the mutation data and is specific to the p53 mutation database, included in the package. If you want to use the program to analyze another data file, you may have to make changes to this routine, or otherwise change the headings in your data file. Right now the following constants index the fields used by the program. For example CodonNumber is in column 4 (= 3 + 1).

```
enum {Type=1, ExonIntron=2, CodonNumber=3, CodonFrom=6, CodonTo=7,
      AAFrom=8, AATo=9};
```

The other thing that might be useful is the variable filterdata in the routine. This is a switch with the default 0 meaning using all records without filtering. If you change it to 1, the program will ask you for a searchstring and then you can type in “liver”. The program will then use only those records that have the search string “liver” somewhere in that record. This might be useful for fitting model M8 of fig. 1.

(C) The variable Small_Diff might be changed to see whether it helps with the SE calculation.

Version history

V1.0 (1 March 2004) implements the model described in Yang et al. (2003).

V1.1 (30 September 2006) I changed the input data format to read version 10 of the p53 database. I also added some code to count different types of mutations at each location (codon) in the refseq. Those counts are printed out in the out.txt file and may include complex mutations that are not used in the ML analysis, such as insertions, deletions and mutations that involve 2 or 3 codon positions.

V1.2 (9 December 2011) I changed the code so that it can read an intron-border file, which is used by the program to more precisely identify CpG and di-pyrimidine sites. In V1.0 and 1.1, CpG and di-pyrimidine sites at the intron-exon boundaries may be misidentified. This was implemented at the request of Nina Stoletzki.

Another change in this version is that I use the trinucleotides (triplets) around the changed position to modify the CpG rate and di-pyrimidine sites, while version 1.0 and 1.1 used quintets, as in Yang et al. (2003). Thus the rate for the C→A mutation in “C GCG A” to “C GAG A” is not modified using the CpG rate ratio parameter in V1.1, but is modified in V1.2.

References

Yang, Z., S. Ro, and B. Rannala. 2003. Likelihood models of somatic mutation and codon substitution in cancer genes. Genetics 165:695-705.