A quick guide to PartitionFinder

Rob Lanfear, June 2011

PartitionFinder is a Python program for simultaneously choosing partitioning schemes and models of molecular evolution for DNA sequence data. It is useful to use before running a phylogenetic analysis of DNA sequence data, in order to decide how to divide up your sequence data into separate blocks before analysis, and to simultaneously perform model selection on each of those blocks.

Quick Start

Have a look at the example file to get your input files in order.

1. Open Terminal and cd to the directory with PartitionFinder in it
2. Run PartitionFinder by typing at the command prompt:
   `python PartitionFinder.py –p -1 <foldername>`
   where '<foldername>' is the full file path to a folder containing a Phylip alignment and a PartitionFinder configuration file called 'partition_finder.cfg'. E.g.
   `python PartitionFinder.py –p -1 /Users/Rob/new_analysis`

The results will be in new subfolder called "analysis" within the original folder. The most important files are "best_schemes.txt" and "all_schemes.txt".

Before You Start

Make sure you have Python 2.7 or later installed. To do this, open Terminal and type "`python`". It will tell you the version you have. If you have anything before 2.7, update python using the appropriate installer from here: http://www.python.org/getit/

Input Files

There are two input files, a Phylip alignment and a configuration file.

Alignment

Your DNA alignment needs to be in Phylip format. We use the same version of Phylip that PhyML uses, which is described in detail here http://www.atgc-montpellier.fr/phyml/usersguide.php?type=phylip

Configuration File

This file should always be called "partition_finder.cfg". The best thing to do is to base your own .cfg on this example file. Most of it should be self explanatory, but in case you want an exhaustive list of everything you can and can't do, here it is.

alignment: the name of your DNA sequence alignment. If it's not in the same file as the .cfg file, you will need to supply a full filepath here (e.g. "`/Users/Rob/Desktop/new_analysis/alignment.phy`"), otherwise just the filename is fine (e.g. "`alignment.phy`").

models: This setting controls the models used in model selection for each of your partitions. It can either be 'all' (to use all 56 models), or a comma-separated list of models to use any subset of the 56 models (e.g. "GTR+G, GTR+I+G").

PartitionFinder performs model selection on each partition, so that in your results it not only tells you the best partitioning scheme, but also which model of molecular evolution is most appropriate for each partition in your dataset. In that way, PartitionFinder is the only program you need to use before your phylogenetic analysis, and you don't need to do any further model selection after PartitionFinder is done.

If **models** is set to 'all' then model selection will be performed on a list of the usual 56 sub-models of the GTR+I+G model. These comprise the usual 12 models of molecular evolution (JC, K80, TrNef, K81, TVMef, TIMef, SYM, F81, HKY, TrN, K81uf, TVM, TIM, and GTR), each of which comes in four flavours: on its own, with invariant sites (+I), with gamma distributed rates across sites (+G), or with both gamma distributed rates and invariant sites (+I+G).

If you want to restrict the list of models considered, you can do that by specifying any particular list of models from the above 56 (use the terminology as given above). For instance, if you were intending to use RaxML to do your final phylogenetic analysis, you will note that RaxML only implements either the GTR+G or GTR+I+G models. There would be little point making PartitionFinder analyse all 56 models of molecular evolution on all possible partitions if in your final analysis you will be restricted to choosing between two models. To deal with this, simply specify the models you do want to use in the **models** variable. In this case you would write:

```
models = GTR+G, GTR+I+G
```

[partitions]: The lines following this statement define the starting partitions for your analysis. PartitionFinder will never suggest additional sub-divisions of these partitions, it will just tell you which of them you could/should join together according to various information-theoretic measures.

Under [partitions] you should define a single partition on each line. These partitions do not have to include all of the sites in your alignment (although you'll get a warning if they don't), but they cannot be overlapping. I.e. each site in the alignment can only be included in a single partition.

The format for designating partitions is very similar to that used in most other phylogenetics packages. You can specify a single site in the alignment, a range of sites, or a range of sites with a repeat value. Examples of these are shown below:

```
Single_site_partition = 1
Range_of_sites_partition = 2-999
Third_codon_positions = 3-999\3
```

You can have semi-colons at the end of each line or not, as you wish. Hopefully this helps a little if you're cutting and pasting from nexus files.

search: which partitioning schemes you want to analyse in your search for the best scheme, either 'all' (analyse all schemes) or 'user' (analyse only user-defined schemes). In general 'all' is only practical for analyses that start with 10 or fewer partitions defined (see below).

Whether you can analyse all schemes will depend to some extent on how much time you have, and to a greater extent on what is computationally possible. If you have any more than 12 partitions to start with you should not choose 'all'. This is because the number of possible schemes for 13 partitions is almost 28 million. This number goes up VERY fast too, e.g. for 16 partitions the number of possible schemes is over 10 billion. It's just not possible to analyse that many schemes exhaustively. For 12 partitions, the number of possible schemes is about 4 million, so it might be possible to analyse all schemes for very small datasets if you have time to wait, and a fast computer with lots of processors.

If you can't (or don't want to) analyse all schemes, set **search** to 'user'. You can then define the schemes you wish to compare, and give each one a name. To define a scheme, simply use parentheses to join together partitions that you would like to combine. For instance, if I had defined three partitions for a protein-coding gene ('1st_pos', '2nd_pos', and '3rd_pos'), I might want to compare the following schemes: (i) all partitions analysed separately; (ii) 1st and 2nd codon positions are analysed together, but 3rd positions separately; (iii) all partitions analysed together. I would do this as follows, with one scheme on each line:

1_2_3 = (1st_pos) (2nd_pos) (3rd_pos)
12_3 = (1st_pos, 2nd_pos) (3rd_pos)
123 = (1st_pos, 2nd_pos, 3rd_pos)

Running PartitionFinder

Once you have your input files set up, to run PartitionFinder you:

1. Open Terminal and cd to the directory with PartitionFinder in it
2. Run PartitionFinder by typing at the command prompt:
   ```
   python PartitionFinder.py –p -1 <foldername>
   ```
   where '<foldername>' is the full file path to a folder containing a Phylip alignment and a PartitionFinder configuration file called 'partition_finder.cfg'. E.g.
   ```
   python PartitionFinder.py –p -1 /Users/Rob/new_analysis
   ```

The "–p -1" is optional – it will make sure that PartitionFinder uses all the available processors on your computer. If you remove it, PartitionFinder will use a single processor, or you can set it to any number to control how many processors will be used.

There are a couple of additional commandline options that can be useful:

--force-restart delete all previous output and start afresh. Useful if you have had to quit an anlaysis halfway through, and the program won't restart.

-p N, --processes=N Number of concurrent processes to use. Use -1 to match the number of cpus on the machine. The default is to use 1.

Output files

All of the output is contained in subfolder of the folder you supplied to PartitionFinder, called 'analysis'. There is a lot of output, but in general you are likely to be interested in four things, maybe this order:

best_schemes.txt: has information on the best partitioning schemes found, according to different information-theory metrics. The Akaike Information Criterion (AIC), the corrected Akaike Information Criterion (AICc) and the Bayesian Information Criterion (BIC). Often these will return the same answer. When they don't I have found in initial simulations that the BIC returns the best answers. This is the subject of ongoing research…

all_schemes.txt: contains the same information as best_schemes.txt, but in csv format, and for all schemes that were compared. This is probably only useful if you're interested in working on methods of finding good partitioning schemes. But, if you subscribe to the view that you should choose the least parameterised model within a certain number of AIC units from the best possible model, then this will be useful to you too.

subsets folder: contains detailed information on the model selection performed on each partition, or combination of partitions (which we call a 'subset'). This output is very similar to what you would get from any model-selection program. Each model tested is listed, in increasing order of AIC score. There will be some minor differences from other model selection programs, due to the implementation we use, but the results should be broadly similar.

schemes folder: contains detailed information on all schemes analysed in a separate .txt file for each scheme. Most of this information is contained in all_schemes.txt, apart from the results of the model-selection on each partition in a scheme.

Citation

There isn't one yet.