Sep 12, 2006

Pyraxml2

(c) Frank Kauff, Duke University, Dept of Biology, PO 90338, Durham, NC 27708, USA; fkauff@duke.edu

NOTE: The current version of pyraxml2 and this manual (Sep 12, 2006) are beta and under development. Your comments are welcome. Please send bugs, problems, suggestions to fkauff@duke.edu.

About Pyraxml2

Pyraxml2 is a wrapper for RAxML-VI-HPC version 2.1 (raxml). The main purpose is to make the use of raxml easier for those of us that use data sets in NEXUS format. Pyraxml2 can read NEXUS data files and converts them to the relaxed PHYLIP format which raxml requires. Moreover, it can handle the exclusion of characters and deletion of taxa prior to converting these files. Pyraxml2 reads all options and command parameters from your NEXUS file, and prepares the necessary input files and command-line options for raxml. It can also prepare the necessary scripts for MPI parallel environments. Pyraxml2 supports all major options of RaxML-VI-HPC version 2.1.0. To efficiently use pyraxml2 and understand the options and parameters, you need to be familiar with raxml and the NEXUS file format. This manual does not teach you how to analyze your data with raxml.

Installation

pyraxml2 is written in python. It needs version 2.4 of Python (or higher). Python2.3 might work as well, but I haven't tried.
Pyraxml2 also needs biopython installed (www.biopython.org). Most importantly, a fairly new version of it. The current (08. Aug 2006) official version 1.42 does not have the necessary changes incorporated yet. If, at the time of reading this, biopython is at version 1.43 or higher, you're fine. Otherwise you will need to download the cvs version at
http://cvs.biopython.org/cgi-bin/viewcvs/viewcvs.cgi/biopython/biopython.tar.gz?tarball=1&cvsroot=biopython

If this link causes problems, just go to www.biopython.org, click on CVS, then on 'Biopython CVS home', then 'biopython', and then the 'download tarball' button at the bottom of the page. The download of biopython should start automatically.

Biopython will tell you that it needs the egenix tools package for python. It is available at http://www.egenix.com/files/python/eGenix-mx-Extensions.html#Download-mxBASE . Either choose

the specific installer for your system, or download the source package for all platforms. In the latter case, you would install them like any other python package (you need root access to your computer):

```
tar -vxzf egenix-mx-base-2.0.6.tar.gz
cd egenix-mx-base-2.0.6
sudo python setup.py install
```

Usually, biopython installs quickly and easily. For standard installation, you need root access to your computer. After downloading the biopython tarball, do

```
tar -vxzf biopython.tar.gz
cd biopython
sudo python setup.py install
```

If this doesn't work, please refer to the BioPython manual in order to find out what's going wrong...

Pyraxml2 is a commandline program and can be started by entering

```
python pyraxml2 nexus_input [options]
```

Depending on your shell / commandline set-up, you may just simply type

```
pyraxml2 nexus_input [options]
```

or

```
./pyraxml2 nexus_input [options]
```

Pyraxml2 needs to know where your raxml-binaries are installed. If you open pyraxml2 in your text editor, in the beginning of the script the two lines that start with 'RAXML=' and RAXML_MPI=' (currently lines 32 and 33) contain the path to your raxml executables. Adjust as necessary.

Commandline

Pyraxml2 has a few commandline-options

-x : clean-up raxml output files from previous raxml runs. Raxml usually complains if output files from previous runs are already present. The -x option deletes these file before starting raxml. CAREFUL: This option just deletes everything that follows the pattern RAxML_*

-n : Converts a newick tree file to a nexus tree file. Works like

```
pyraxml2 -n <newick_input >nexus_output
```

-h : gives a little help.

Nexus

All other stuff dealing with actual raxml commands is now in a raxml block in the nexus file:

```
begin raxml;
           command;
           command;
           ...
end;
```

NEXUS file parsing is implemented using the biopython nexus parser. Syntax follows the official NEXUS standard as in Maddison et al. (1997). However, identifiers (taxon names, set names, etc.) are case sensitive, and ' ' (space) and '_' (underscore) are not equivalent. All other NEXUS features are supported, e.g. multi-line and nested comments. If pyraxml2 doesn't read your file, make sure that it is proper NEXUS. Many files out there that claim to be, are not. As a rule of thumb, if Paup, Mesquite or McClade can read it, it's ok. If those programs (written by the founders of the NEXUS standard themselves) complain, something is probably wrong.

The raxml block can be anywhere in the NEXUS file. Raxml NEXUS blocks know the following commands and options:

exclude

`exclude` *`character list | character-set name`*

Excludes a list of characters from the analysis. It accepts a list of characters as in

```
exclude 5 3 17-29 8-100\3;
```

or a list of previously (in a nexus sets block) defined character sets

```
exclude codon3 gene1;
```

or a mixture of both

```
exclude 1-500 codon3;
```

Note: If later you define partitions to run raxml with multiple models, the partitions will be adjusted automatically to reflect that characters have been excluded.

delete

```
delete taxon list | taxon-set name
```

Same as `exclude`, just with taxa.

startingtree

```
startingtree filename = filename tree = newick tree description |
     tree-name | random | parsimony
```

The option `filename` accepts the filename with the startingtree, which can be either in nexus or in newick format. The `tree` option either takes the name of a previously (in a nexus trees block) defined tree, or an inline newick tree, as in

```
startingtree tree=mytree;
```

or

```
startingtree tree=(a,(b,(c,d)));
```

The `tree` option also knows `random` and `parsimony`, which specify a random starting tree or the (default) parsimony starting tree.

constraint

```
constraint filename = filename tree = newick tree description | tree-
     name
```

Same syntax as startingtree. Pyraxml2 finds out whether the tree is bifurcating or multifurcating and uses the -q / -r options of raxml accordingly.

weight

```
weight filename = filename
```

Specifies the filename with the weights for raxml, analogous to raxml option -a.

partitionmodels

```
partitionmodels partition = partition_name part1 = matrix1 [part2 =
     matrix2 ...]
```

For AA data sets, raxml supports to apply different transition matrices for different partitions. With the partitionmodels command, you specify first the name of the character partition, and then a specific transition matrix for each of the subpartitions.
Example: if your character partition definition looks like

```
begin sets;
     charpartition genes = lsu: 1-400, ssu: 401-800, rpb1: 801-1000;
end;
```

then you can define matrices like:

```
partitionmodels partition=genes lsu=JTT ssu=WAGF rpb1=BLOSUM62 ;
```

Please keep in mind that partition names are case sensitive, so rpb1 is different from RPB1.

mapbipartitions

```
mapbipartitions filename = filename tree = newick tree description |
     tree name bipartitions=filename format = newick | nexus
```

Implements the -f b option of raxml, which uses a file with bootstrapped trees to map the proportions of the bipartitions onto a given topology. The topology is specified as a file or a given tree (with the `filename` or `tree` options, resp.), and the option `bipartitions` specifies the bootstrap trees file. It can either be in nexus or plain newick format, which is indicated with the `format` option.

```
mapbipartitions filename=my_best_tree.tree
     bipartitions=raxml_bootstrap_trees format=newick ;

mapbipartitions tree=tree1 bipartitions=bootstrap_paup.trees
     format=nexus ;
```

raxml

```
raxml model = GTRxxx | PROTxxx
          partition = partition-name
          epsilon = likelihood epsilon
          algorithm = d | e | b
          ncat = number of categories
          intermediate = yes | no
          rearrangements = initial rearrangement setting
          bootstrap_seed = -1 | random seed value
          nreps = number of replicates
```

Most of these options correspond straightforwardly to their raxml counterparts, such as `epsilon` (-e), `algorithm` (-f), `ncat` (-c), `intermediate` (-j), `rearrangements` (-i), `nreps` (-#), `model` (-m). The `bootstrap_seed` options specifies the random number generator seed and activates the bootstrapping, as in the raxml-option -b. However, when specifying -1 as seed, the actual seed is

generated by a random number itself.
`Partition` gives the name of a character partition, which needs to be defined in a sets block (not in the raxml block!) using the NEXUS `charpartitions` command. Pyraxml2 then writes the proper input file for raxml. The definition of a character partition follows the official nexus standard (e.g., as in paup):

```
begin sets;
     charpartition genes = lsu: 1-400, ssu: 401-800, rpb1: 801-1000;
     charpartition codons = lsu: 1-400, ssu: 401-800,
          pos1: 801-1000\3, pos2: 802-1000\3, pos3: 803-1000\3;
end;
```

In the raxml command, you can then simply use the partition with

```
raxml partition=codons;
```

and raxml would use four different models for the search. If you exclude characters with the `exclude` command, the partitions are automatically adjusted. Example

```
exclude 1-200;
raxml partition=codons;
```

would result in multiple-model file for raxml that looks like:

```
ssu = 1-200
lsu = 201-600
pos1 = 601-800\3
pos2 = 602-800\3
pos3 = 603-800\3
```

because the partition boundaries in this example were moved 200 characters to the start.

mpi

```
mpi pe=parallel_environment ncpus=number_of_cpus
```

These parameters are actually not for raxml, but for the MPI parallel environment settings. ncpus specifies the numbers of cpus to be used, and pe specifies the parallel environment. This is equivalent to the -pe option of qsub or your qsub script, and depend on your MPI architecture. If you don't know what values to put there, ask your system administrator.

Unsupported options:

-h, -v and -y are currently not supported.

Output

Newick tree files produed by raxml are automatically converted to NEXUS tree files.

Example

A typical nexus file, say `example.nex`, could look like this:

```
begin data;
... here comes your data matrix, etc....
end;

begin trees;
tree my_backbone = (outgroup,(a,b,c,d),(e,f,g),(h,i,j,k));
end;

begin sets;
charset introns = 30-234 130-483;
charpartition genes = g1:1-200, g2:201-511, g3:512-897;
end;

begin raxml;
     exclude introns;
     constraint tree=my_backbone;
     mpi pe=high_priority ncpus=50;
     raxml model=GTRGAMMA partition=genes
          epsilon=1.4 ncat=35 rearrangements=15
          bootstrap_seed=-1 nreps=500;
end;
```

Starting this file (assumed you have completed the data block) with

```
pyraxml2 example.nex
```

would write and submit an mpi script for your cluster, instructing raxml to partition your data into three parts, and do 500 bootstrap replicates with a constraint tree and the introns excluded from the analysis.

Usage

Single cpu jobs

If you use pyraxml2 for a single cpu job (no mpi command in your raxml block), it works like starting raxml itself. That means, if you start pyraxml2 on the commandline, your machine will start searching for a tree, or doing a bootstrap, or whatever it is you ask raxml to do. If instead you want to submit it as a job to a cluster, you can do it the same way as you do with any other program (most likely by embedding the pyraxml2 call into a submission script for your cluster).

Parallel MPI jobs

If you have the `mpi` command in your raxml block, pyraxml2 works slightly differently. Most likely, what you want is to submit the parallel job to a cluster. Therefore, pyraxml2 creates the necessary submission script, submits it to the cluster, and exits. The submission script in the source code of pyraxml2 may need adjustment to your local cluster environment.

Example Files

Pyraxml2 comes with five example NEXUS files in the folder *examples*. They demonstrate the use of some of the options, each one comes with a short explanation about what it does. Example no. 4 (example4_mapbipartitions.nex) needs the result of example1 and example2, so better execute example1 and example2 before you try out example4.

References

Maddison DR, Swofford DL, Maddison WP (1997) NEXUS: An extensible file format for systematic information. Systematic Biology,**46**,590-621.

Distribution