A Brief Introduction to the Phylogenetic Analysis Library Version 1.5

Matthew Goode [1], Korbinian Strimmer [2], Alexei Drummond [3], Ed Buckler [4], Allen Rodrigo [1]

[1] School of Biological Sciences
University of Auckland
Private Bag 91029, Auckland, New Zealand
m.goode@auckland.ac.nz, a.rodrigo@auckland.ac.nz

[2] Department of Statistics
University of Munich
Ludwigstrasse 33, 80539 Munich, Germany
strimmer@stat.uni-muenchen.de

[3] Department of Statistics and Department of Zoology
Oxford University
OX3 1PS United Kingdom
alexei.drummond@zoology.oxford.ac.uk

[4] Institute for Genomic Diversity,
Cornell University
159 Biotechnology Bldg, Ithaca, NY 14853-2703
esb33@cornell.edu

Abstract

The Phylogenetic Analysis Library (PAL) is a Java class library for performing phylogenetic inference and analysis of primarily molecular sequence data. In this paper the features of PAL are described, in particular the new features since version 1.1. Practical examples of the usage of PAL are also given.

Keywords: "Phylogenetic analysis" Java "Open Source"

1 Introduction

The Phylogenetic Analysis Library (PAL) project (Drummond and Strimmer, 2001) is a collaborative, open-source project developed under the lGPL license (see www.gnu.org). The purpose of PAL is to provide a useful toolkit of compatible data structures and methods for molecular sequence analysis. PAL aims to provide well written implementations of standard phylogenetic tools. Features include tree construction/manipulation, simulation, and statistical analysis. PAL is written using the popular Java programming language (see java.sun.com) and is designed with a strong object-orientated focus. Java is a scalable modern language which allows the easy construction of cross-platform applications.

The intended audience for the PAL project is two-fold. First, PAL is a tool for the developers of phylogenetic applications. Second, PAL aids the bioinformatics researcher with ambitious experimental ideas that require a more programmatic approach than the user interface or scripting abilities of other packages. The PAL resource has been utilised in a number of projects including the Mesquite project (http://mesquiteproject.org), the DNA Surveillance project (Ross *et al*., 2003), and others. The PEBBLE application (http://www.cebl.auckland.ac.nz) provides in part a graphical front-end to PAL, and the VANILLA package (see link from PAL website) provides command line access.

1.1 Availability

The PAL project website has moved a number of times in recent years. The current location, stable for the foreseeable future is at:

http://www.cebl.auckland.ac.nz/pal-project

All releases of PAL are archived and available from the website. Snapshots of the current development version are posted semi-weekly. An email list is available to receive announcements regarding new versions.

1.2 Contributing

As PAL is open source and released under the lGPL licence users are free to make changes for their own

Package	Brief Description
pal.alignment	Data structures, and utilities for sequence alignment.
pal.algorithmics	A generic set of components for algorithmic procedure such as hill climbing and simulated annealing.
pal.coalescent	Modelling of population genetic processes using coalescent theory.
pal.datatype	Classes and tools for describing sequence data types.
pal.distance	Data structures and tools related to generic distances.
pal.eval	For evaluating evolutionary trees and estimating parameters under a likelihood framework.
pal.gui	Tools for the display of phylogenetic trees, and other special objects.
pal.io	Text input/output tools.
pal.math	Optimisation methods, special functions, numerical derivatives, etc.
pal.misc	Classes and tools that do not fit into other packages.
pal.mep	Models relating to measurably evolving populations.
pal.popgen	Linkage disequilibrium tools.
pal.statistics	Tree tests, distributions, bootstrap estimators.
pal.substmodel	Models of substitution, with common nucleotide models (GTR), amino acid models (Blosum62, Dayhoff), and codon models.
pal.supgma	Classes specific to the sUPGMA analysis.
pal.tree	Tools and data structures for building, describing, and modifying trees.
pal.treesearch	Classes relating specifically to tree search methods, such as ML search.
pal.util	Utility methods, such as sorting.
pal.xml	Preliminary support for XML input/output.

Table 1: The 19 Java Packages defined in PAL (version 1.5). New packages since version 1.1 are: pal.algorithmics, pal.mep, pal.popgen, pal.supgma, pal.treesearch, and pal.xml.

personal use. Users who have developed a worthwhile addition to PAL, and who wish to share with other users, can submit their changes to the PAL maintainers by following the related information found on the website. Users who contribute a substantial amount can apply for direct access to the PAL CVS[1] repository.

2 The Power of PAL

The last paper relating to PAL, Drummond and Strimmer (2001), described the features of PAL version 1.1. The most recent release of PAL is version 1.5 (as of November 2003), and a number of new features have appeared. There are now over 250 classes spread across 19 packages (see Table 1).

2.1 Original Features

A selection of specific features of PAL available since version 1.1 includes:

- The reading and writing of sequence alignment, distance matrices, and phylogenetic trees.
- The modelling of substitution for nucleotide and amino acid sequence data, with rate heterogeneity (Yang 1994).
- Simulation of coalescent intervals, and estimation of demographic parameters (Donnelly and Tavare, 1995).
- Forward simulation of sequence data across a tree.
- Estimation of demographic parameters.
- Statistical tests such the Kishinio-Hasegawa and Shimodaira-Hasegawa tests (Goldman *et al*., 2000).
- Tree construction methods via cluster methods such as neighbour-joining, UPGMA, and sUPGMA (Drummond and Rodrigo, 2000).

This is not an exhaustive list, and the reader is again referred to Drummond and Strimmer (2001).

[1] CVS, or Concurrent Versioning System is a tool for managing concurrent access to a software project. See www.cvshome.org.

2.2 New Features

As of version 1.5 a number of new features have appeared in PAL, including:

- Maximum likelihood tree search for unconstrained, unrooted trees. Includes advanced features such as support for simulated annealing, simultaneous NNI (Guindon and Gascuel, 2003), and simultaneous substitution model estimation.
- Easier access for fixed topology ML analysis, and faster, more general, likelihood calculation.
- The serial-sample analysis of (Drummond *et al.*, 2001), and stronger support for sUPGMA (Drummond *et al.*, 2000).
- The neutral and selection codon models of substitution of Neilson and Yang (1998).
- More stable input/output, include limited support for Nexus alignment files.
- More comprehensive data type conversion.
- More robust and powerful tree manipulation including mid-point rooting, rooting by an out-group, taxa restriction.
- The inclusion of "Tool" classes which provide static access to common features. PAL has an ever adapting interface, and the new Tool classes are a step to providing a stable access point across versions of PAL.
- Additional hierarchical clustering methods of wUPGMA, single-linkage, and complete linkage.
- Linkage disequilibrium calculation and display.
- Easier bootstrap analysis.
- Numerous new utility functions.

3 Examples of Usage

Given a brief overview of the functionality, the reader may be left pondering exactly how easy PAL is to use. An extensive and powerful toolkit is lacking if the tools are too hard to use by the average user.

An exhaustive discussion of PAL usage is beyond the scope of this paper. Instead a few simple examples based around phylogenetic tree inference using maximum likelihood and neighbour-joining are given to provide the user with a feel for the applicability and approach of PAL. An understanding of the basics of Java programming may be useful, but the purpose is mainly to illustrate the simplicity of usage. Related source code, such as package imports, are not shown.

3.1 Housekeeping

The Alignment object is a PAL data structure for representing a sequence alignment. SubstitutionModel objects represent a model of character substitution over time. The later examples in this section rely on the existence of Alignment and SubstitutionModel objects, so a good starting point would be the creation of such objects. As neighbour-joining also requires the use of a distance matrix a process for construction of such a matrix, based on an alignment and a substitution model, is also described.

3.1.1 Loading an Alignment

Sequence alignment formats currently supported by PAL are 'Fasta', 'Clustal' and 'Phylip' formats, or any format that simply lists the aligned sequences interleaved or sequentially without major annotation. Limited support exists for loading a Nexus alignment. Basic alignment input can be accessed from the AlignmentTool class. To load an alignment, the following could be used:

```
FileReader fr = new FileReader("example.clustal");

DataType dt = DataTypeTool.getNucleotides();

Alignment base =
        AlignmentTool.readAlignment( fr, dt );
```

In general a DataType object is also required in case the alignment format does not dictate the residue type. In the previous example the DataTypeTool class was used to create a DataType object representing nucleotide characters.

3.1.2 Constructing a Substitution Model

There are several models of substitution defined in PAL, including nucleotide, amino acid, and codon models. PAL makes a distinction between a rate matrix, and a substitution model. A substitution model encompasses additional functionality such as rate heterogeneity and may utilise one or more rate matrices. A user would normally create a rate matrix, such as a GTR matrix (Lanave, *et al.* 1984), and then create a substitution model around such a matrix. For the average user, the SubstitutionTool class hides such underlying detail. To construct a general time reversible nucleotide model the following could be used:

```
double[] freqs =
        AlignmentUtils.estimateFrequencies( base );

double a = 4; double b = 1; double c = 3;

double d = 10; double e = 2;

SubstitutionModel gtr =
        SubstitutionTool.createGTRModel(
                a, b, c, d, e, freqs
        );
```

3.1.3 Calculating Evolutionary Distances

Clustering methods, such as UPGMA and neighbour-joining, rely on a distance matrix, and do not work directly with alignment data. The evolutionary distance is a useful distance measure. The evolutionary distance between two sequences is taken as the branch length of the maximum likelihood tree containing only the two sequences, under a set model of substitution.

Using the DistanceTool class the creation of a distance matrix based on evolutionary distances is trivial:

```
DistanceMatrix dm =
        DistanceTool.constructEvolutionaryDistances(
                base, gtr
        );
```

3.2 Tree Construction

Given that an alignment has been obtained and a substitution model created (along with a distance matrix if required) it becomes possible to construct a phylogenetic tree. Two methods of tree construction are described here: the powerful, yet computationally intensive method of maximum likelihood tree search, and the fast, but less perhaps less accurate, neighbour-joining method.

3.2.1 Maximum Likelihood Tree Search

A powerful and robust tree building method is that of maximum likelihood (Felsenstein 1981). The purpose of the tree search algorithm is to find the topology and branch lengths of a tree that maximise likelihood, given a sequence alignment. A model of substitution is important. The search can also include the substitution model parameters. The user is warned that, even though much effort has been expended to achieve efficient performance, a maximum likelihood tree search can take a substantial amount of time. The user should also be aware that heuristic tree search/optimisation methods such as used by PAL are not guaranteed to always find the true maxima, and multiple runs are always recommended.

The tree search algorithm used by PAL has a number of user changeable parameters. To utilise predefined general purpose search parameters (as determined by PAL developers), the TreeSearchTool class can be used. For example, given the existence of the alignment and substitution model defined previously the following may be used:

```
Tree t = TreeSearchTool.doUnrootedSearch(
                base, gtr

        );
```

To include the model parameters in the search use the following:

```
Tree t =
        TreeSearchTool.doUnrootedSearchWithModel(
                base, gtr
        );
```

After a search involving model estimation, the model will be updated to the parameters relating to the maximum likelihood tree and substitution model combination.

Advanced fine tuning of the search algorithm is beyond the scope of this document, but are readily adaptable for the user that is familiar with the underlying theory.

3.2.2 Neighbour-joining

The maximum-likelihood tree search can be infeasible when the dataset is too complex (long and many sequences). In such cases the popular technique of neighbour-joining (Saitou and Nei 1987) can be utilised.

Given a distance matrix has been created, as previously described, the construction of a tree using the neighbour-joining method is trivial. One way to accomplish this is by constructing a NeighborJoiningTree object,

```
Tree t = new NeighborJoiningTree( dm );
```

Alternatively the TreeTool class may be utilised,

```
Tree t = TreeTool.createNeighbourJoiningTree( dm );
```

3.3 Finishing Moments

Given the construction of a tree using whatever method, it would be useful to save such a tree to a file for later usage, or exporting to another application. Before saving though, it may be advantageous to root the tree.

3.3.1 Tree Rooting

Both the neighbour-joining method and the maximum likelihood search construct an un-rooted tree. A user wishing to root such trees can use the TreeTool class:

```
Tree rooted = TreeTool.getRooted( t, outgroupNames );
```

Alternatively, if no out-group is defined, the mid-point rooting technique can be employed. Mid-point rooting places the root the tree at the point between the two most distant taxa, and can be performed using the following:

```
Tree midPointRooted =
        TreeTool.getMidPointRooted( t );
```

3.3.2 Saving a Tree

The process of saving a tree is again straightforward. The TreeTool class provides an easy access to tree input and output methods. The following can be used to save a tree in the commonly used Newick format:

```
FileWriter fw = new FileWriter("tree.phy");

TreeTool.writeNewick( t, fw );
```

4 Future Work

The PAL project, while an effective tool in the current version, has many windows of opportunity for future growth. This includes, amongst others, the following:

- New applications of tree search, in particular tree search across constrained trees. There is also the possibility of implementing maximum parsimony.
- A functional MCMC framework for Bayesian analysis of phylogenetic data.
- Improved input/output including improved XML and Nexus support.

- The ability to perform multiple sequence alignment using effective techniques.
- Integration with external libraries. In particular a bridge package is planned between PAL and the BioJava package (see www.biojava.org).
- Advanced graphical representation, especially for the production of publication quality graphics of phylogenetic tree data.
- Unit testing, involving construction of tests to check the integrity of function modules, and for problems introduced by related code changes.
- Improved documentation.
- Super-tree functionality.

Much of the work mentioned will be implemented over time by the major contributors to PAL, but other contributors, are always welcome to add their own direction to PAL.

5 Conclusion

The Phylogenetic Analysis Library is an ongoing and open project to develop a powerful programming toolkit for phylogenetic analysis using the Java language. This article introduced the PAL project, and gave an update on the new features available in version 1.5. Information on availability and contributing was given. Small appetiser examples of the use of PAL were given to illustrate the ease of use, and power, of PAL. Finally, the future work was outlined.

Acknowledgements

Hardware and resource support was in part through NIH grant GM59174. This work was supported by a New Zealand FRST Bright Futures scholarship (A. D.), and Emmy Noether-Fellowship by the Deutsche Forschungsgemeinschaft (K. S.). Current support includes an Allen Wilson Centre of Excellence PhD scholarship (M. G.). The authors wish to thank Greg Ewing, Iain Milne, Bruno Afonso, Steven Woolley and others for their useful contributions and insights.

References

Donnelly, P. and Tavare, S. (1995): Coalescents and genealogical structure under neutrality. *Annu. Rev. Genet.* **29**:401-421

Drummond, A. and Rodrigo, A. G. (2000): Reconstructing genealogies of serial samples under the asuumption of a molecular clock using serial-sample UPGMA. *Mol. Biol. Evol.* **17**:1807-1815.

Drummond, A., Forsberg, R. and Rodrigo, A. (2001): Estimating stepwise changes in substitution rates using serial samples. *Mol. Biol. Evol.* **18**:1365-1371.

Drummond, A. and Strimmer, K. (2001): PAL: an object-orientated programming library for molecular evolution and phylogenetic. *Bioinformatics* **17**(7):662-663.

Felsenstein, J. (1981): Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Mol .Evol.* **17**:368-376.

Goldman, N., Anderson, J.P. and Rodrigo, A.G. (2000): Likelihood-based tests for topologies in phylogenetic. *Syst. Biol.* **49**:652-670.

Goldman, N. and Yang, Z. (1994): A codon-based model of nucleotide substitution for protein coding DNA sequences. *Mol. Biol. Evol.* **11**(5):725-736.

Guindon, S. and Gascuel, O. (2003): A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology* **52**(5):696-704.

Lanave, C., Preparata, G., Saccone, C., and Serio., G. (1984): A new method for calculating evolutionary substitution rates. *Journal of Mol .Evol.* **20**: 86-93.

Lio, P. and Goldman, N. (1998): Models of molecular evolution and phylogeny. Genome Res., **8**:1233-1244.

Matsumoto, M. and Nishimura, T. (1998): Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. Modeling Comput. Simul.* **8**:3-30.

Neilson, R., and Yang, Z. (1998): Likelihood models for detecting positively selected amino acid sites and applications to the HIV-1 Envelope Gene. *Genetics* **148**: 929–936.

Saitou, N., and Nei, M. (1987): The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**(4):406-425.

Rambaut, A. (2000): Estimating the rate of molecular evolution: incorporating non-contemporaneous sequences into maximum likelihood phylogenies. *Bioinformatics* **16**:395-399.

Ross, H.A. , Lento, G.M., Dalebout, M.L., Goode, M., Ewing, G., McLaren, P., Rodrigo, A.G., Lavery, S. and Baker, C.S. (2003): DNA Surveillance: web-based molecular identification of whales, dolphins and porpoises. *Journal of Heredity* **92**:111-114.

Yang, Z. (1994) Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *Journal of Mol. Evol.* **39**:306-314 .