Bio::CDAT – An Object for Evolutionary Analysis of Data

Summary

We propose a Bio::CDAT (**C**haracter **D**ata **A**nd **T**rees) module to facilitate comparative analysis using evolutionary methods by 1) managing evolutionary relationships (by linking data to trees) and 2) allowing coordinated analysis of different types of data (by implementing a generic concept of "character-state" data). Bio::CDAT would take advantage of existing BioPerl objects, include the functionality of Bio::Phylo, and provide the framework to develop interfaces to analysis tools (phylogeny inference, evolutionary rate models, functional shift inference, etc), as well as to file formats and visualization methods appropriate for such analyses.

Background

Much of bioinformatics is a kind of comparative analysis— making inferences from similarities and differences in proteins, regulatory regions, expression patterns, SNPs, and so on — for which evolutionary theory provides the natural theoretical framework. Typically this framework allows one to convert comparative-analysis questions into theoretically well posed questions about transitions in the states of characters over time, according to a "character-state data model" (Figure). In this model the observable features of the "**OTU**s" (Operational Taxonomic Units) are the "states" of a set of underlying "characters". In classical systematics, the OTUs are species whose morphology, behavior, and life-history features are broken down into discrete or continuous states of underlying characters. In sequence analysis, the mapping of features is accomplished by an alignment: the OTUs are sequences, the characters are alignment columns, and the states are residues. An OTU may have many types of characters. Powerful analysis tools are available to model the evolution of characters, both discrete (e.g., nucleotides, amino acids, presence/absence of features) and continuous (e.g., transcript level, splice site usage).

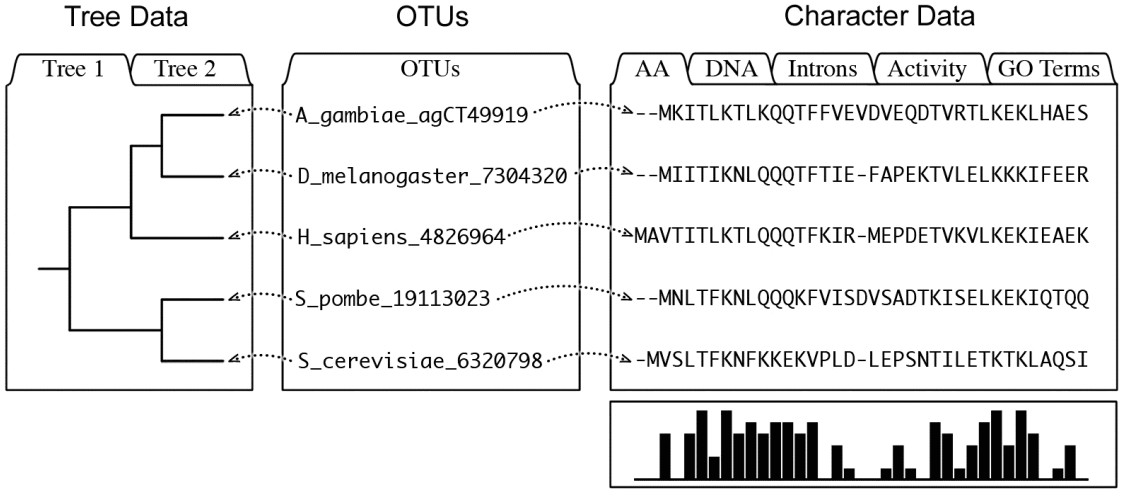

Fig. 1. A CDAT object instance - OTUs are associated with nodes on a tree and with one or more sets of character data.

BioPerl handles many such types of data, as well as trees, but does not provide an interface to data related by a tree, nor does it generalize the concept of character/state data so as to faciliate

data transfer (e.g., via the NEXUS file format, which is far richer than most people know; Maddison, et al., 1997), or to allow generalized interfaces to software for analysis (e.g., PAUP), editing, or visualization (e.g,. Nexplorer).

Of course, all of this could be done without implementing a CDAT object, by taking advantage of Bio::Phylo, so why do it? The reason is simply to make evolutionary analyses easier by facilitating code re-use and interoperability, and to provide a stable interface for BioPerl users (while developers manage the details). The more important it is to do evolutionary analysis, the greater the advantage of having BioPerl methods that:

- Store data sets together with the relevant trees
- Preserve evolutionary relationships among objects even when they are modified
- Integrate different types of data for a set of OTUs into one framework
- Provide generalized interfaces to phylogeny and reconstruction tools
- Integrate inputs and outputs from evolutionary analysis into work-flows

Implementation

In general, we do not want to commit to a specific design without consulting other developers, but two things are relatively obvious:

1. We need a generic interface (Bio::CharMatrix::CharMatrixI?) by which to slice, splice, concatenate, sort, get type, get taxa, etc, for character data, whether the character data are in a sequence alignment object, or in a generic matrix object.

2. We need some kind of "OTU" concept by which to link the tips of trees to rows in character data matrices. It must be possible to link an OTU to many different kinds of character data (e.g., a protein OTU could have an amino acid sequence, a binding coefficient, some GO terms, etc). Its also likely that we would need to link an OTU to multiple trees.

To make the CDAT concept useful, we would need to build up interfaces to external analysis tools. Elements of a generic command language for evolutionary analysis already exist— they are embodied in the NEXUS file format standard and in the interface to tools such as MrBayes, PAUP, HyPhy and so on.

Questions to consider

What are the most important use-cases to support (e.g., function annotation via "RIO" method, detecting positive selection via PAML, protein sequence/structure/function relationships via "evolutionary trace", etc)?

What is the defining feature of a CDAT object? Is it the set of OTUs (i.e., resolve conflicts by insisting that OTU set is same for all trees and data types)?

Where do we foresee difficulties with Bio::CharacterMatrix, and how can these be resolved?

How can an OTU object be implemented to take advantage of existing BioPerl objects?

In many cases we would desire a CDAT object with multiple trees (e.g., a distribution of trees; methodological alternatives). Where is this going to lead to difficulties, and how do we resolve them (e.g., what happens when OTUs are removed from the set)?

Hypothetical example

A hypothetical example that illustrates several ideas is as follows. We have inhibitor data for human kinases and we wish to know which of two trees explains the data better, a tree based on the full domain alignment, or one based on only the binding-pocket residues. Our inputs are a kinase domain sequence alignment in a standard format (e.g., clustalw), a binding-pocket mask (a vector of 1's and 0's indicating which columns are binding-pocket columns), and a matrix of inhibitor data in comma-separated-value format. We import the data, compute two trees, and then compare the likelihood of the inhibitor data given each tree, using the PHYLIP program "contml".

```
use Bio::CDAT;

my $kinase_cdat = Bio::CDAT->new( -file => 'kinase-seqs.aln', -format => 'clustalw', -label =>
'domain alignment', -type => 'aa' );

$kinase_cdat->AddCharFromMatrix( -label => 'ic50 values', -type => 'numerical', -format => 'csv',
-file => 'ic50-data.csv' );

my $tree = $kinase_cdat->InferTree( -source => 'domain alignment', -method =>'protml');

$kinase_cdat->AddTree( -label => 'domain', -data => $tree );

my $binding_pocket_cdat = $kinase_cdat->SelectColumns( -file => 'binding_pocket.fas' );

my $other_tree = $binding_pocket_cdat->InferTree( -source => 'domain alignment', -method
=>'protml');

$kinase_cdat->AddTree( -label => 'binding pocket, -data => $other_tree );

$kinase_cdat->ComputeLikelihood( -data => 'ic50 values', -usertree => 'all', -method => 'contml'
);
```

Note that, although we created a binding pocket CDAT from $kinase_cdat in order to get a tree, both trees were added to $kinase_cdat and then, in the last step, both trees were used to compute likelihoods (alternatively, we could have computed the likelihoods one tree at a time using separate cdats).

Contacts

Arlin Stoltzfus **stoltzfu@umbi.umd.edu**
Rutger Vos rvosa@sfu.ca
Weigang Qiu **weigang@genectr.hunter.cuny.edu**

References

Maddison, D.R., Swofford, D.L., and Maddison, W.P. 1997. NEXUS: an extendible file format for systematic information. Systematic Biology 46: 590-621.

Swofford, D.L., PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). 1999, Sinauer Associates: Sunderland, Mass.

Gopalan, V., Qiu, W.G., Chen, M.Z., and Stoltzfus, A. 2006. Nexplorer: phylogeny-based exploration of sequence family data. Bioinformatics 22: 120-121.