PhyloProf: a web-based application for the generation of phylogenetic profiles

Daniel S. Standage; Bioinformatics Undergraduate; Brigham Young University, Provo, Utah 84604;
Submitted 10 December 2008.

*Predicting gene relationships and gene functions is a central goal of biology and bioinformatics. Phylogenetic profiles have provided an evolutionary model for gene comparison based on the assumption that genes involved in the same structure or pathway will co-evolve. I here present **PhyloPro**, a web-based application that can be used to compute phylogenetic profiles for any set of genes. **PhyloPro** searches a microbial database for homologs of query genes, and for each gene generates a profile indicating which genomes contain a homolog of the gene. Results can be viewed using the interface or exported to a data file for further analysis.*

INTRODUCTION

Predicting the function of individual proteins and relationships between proteins is one of the central goals of bioinformatics and biology in general. Advances in molecular biology and biotechnology have led to an explosion in the amount of DNA and protein data available today. The magnitude of these data demands automated, computational approaches to DNA and protein analysis.

Phylogenetic profile analysis was presented by Pellegrini et al. (1999) as a method for predicting gene function. A phylogenetic profile is a simple binary string which represents the occurrence or absence of a gene in a set of genomes. This method is based on the assumption that proteins involved in related structural complexes or biochemical pathways will generally evolve together, and the data demonstrated that proteins with similar profiles had a strong tendency to be functionally linked.

Analysis of phylogenetic profiles can lead to meaningful predictions regarding gene functions and relationships. Comparing the phylogenetic profiles of novel proteins to the phylogenetic profiles of well-annotated proteins can facilitate the prediction of these novel proteins' functions. Additionally, given the phylogenetic profile of a protein known to be involved with a disease, phylogenetic profile analysis can predict other genes also involved with that disease based on their phylogenetic distribution.

I present **PhyloProf**, a simple web-based tool for generating phylogenetic profiles. The application computes the distribution of query genes across 771 microbial genomes, making these profiles available for export and further analysis.

METHODS AND MATERIALS

Application Logic

The logic behind the **PhyloProf** application is a program implemented using Perl, freely available Perl libraries (DBI, BioPerl), and the MySQL database management system. When the application receives a request from the interface, the query sequence is stored in the file system and the request is placed on a queue. Server performance is

manage by a Perl script that executes requests in the order they were received, ensuring thatnot too many jobs are running simultaneously.

Phylogenetic profiles are computed by BLASTing each query gene against a database containing all 771 currently sequenced microbial genomes, and accepting all hits below the user-provided e-value cutoff. A binary string of length 771 is returned for each query gene, each 1 in the string representing a microbial genome that contains at least one homolog of the query gene.

Application Interface

PhyloProf provides a web-based interface to a tool for creating gene phylogenetic profiles. This interface was created using a combination of PHP server scripting, XHTML, and CSS. The main page of **PhyloProf** provides a form in which users can submit sequence data for the genes they would like to analyze. The form allows user to specify several parameters, including the BLAST program to be used in the analysis and the e-value cutoff used to determine gene homology.

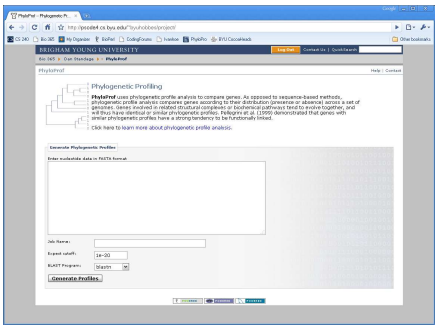

Figure 1: The PhyloProf application provides access to a simple tool for creating phylogenetic profiles.

When a job request is complete, the interface displays the results on a results page. The user may visually analyze his/her data, or they may export it to a data file for further analysis. Currently, this results page will be available to the user indefinitely.

As generating phylogenetic profiles is a computationally intensive task, the interface provides a page to indicating that the user's request is in progress. Included on this page is a link that can be used to check the status of the request at any time, so that users can leave the page and return at their convenience. This page will automatically redirect to the results page when the job request is complete.

RESULTS

The **PhyloProf** application has a very aesthetic user interface and places heavy emphasis on the user experience. The application performs well for small jobs that involve querying the nucleotide database. The application will handle large jobs as well, although the quality of the user experience may decrease as generating phylogenetic profiles is a computationally intensive task. Nonetheless, users may return to retrieve their results at their convenience, assuming those results are complete.

Although **PhyloProf** has a protein database, there have been difficulties integrating this database with the application. Being able to determine the organism to which a hit gene belongs is key to computing a phylogenetic profile. Supplementary files provided by NCBI indicate which accession numbers belong to which organisms, but this appears to be limited to nucleotide data. Although nucleotide data can be useful, protein data is likely more appropriate for phylogenetic profile analysis because it is based on assumptions regarding protein evolution and function.

The analysis features provided by **PhyloProf** are very simple. The results page simply displays each query gene, indicating which genomes contained a homolog of that gene. Any further analysis (such as hierarchically clustering genes or scoring genes based on a weight matrix) must be done by the user by exporting the data file and using loading it into the statistical software of their choice.

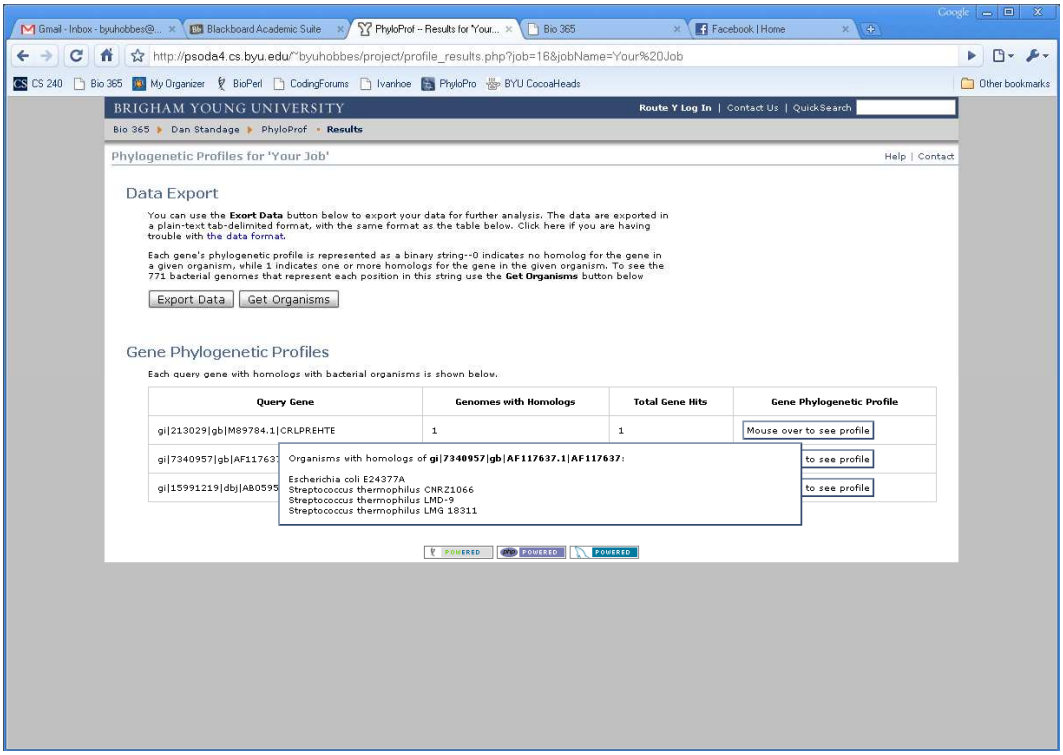

Figure 2: The data analysis features provided by PhyloPro are relatively simple. Extending these analysis features is a primary goal of continued development.

CONCLUSIONS

The **PhyloProf** application is a simple yet powerful tool for gene comparison. I have identified several priorities to guide continued development of the application. Of primary importance is integrating the protein database with the application to provide access to those BLAST programs requiring a protein database. This will require further analysis of the protein data itself and writing some scripts that may take a long time to execute.

Another primary goal is to integrate more tools for data analysis with the application. Scientists with computational experience may have no trouble exporting and analyzing the results themselves, but many scientists will not have the time or experience required to do this. Providing a tool to score and/or cluster gene profiles will go a long way to improve this application.

Overall, this project turned out well and I am very pleased with the results thus far.

REFERENCES

Pellegrini, M. et al. (1999). Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of the National Academy of Sciences, USA*. **96**, 4285-4288.

APPENDIX

Below is the code for the algorithm used to generate phylogenetic profiles.

```
#!/usr/bin/perl

# Import libraries
use DBI;
use Bio::Perl;
use Bio::Tools::Run::StandAloneBlast;
use strict;

# Usage and command line arguments
my $usage = "perl profile_analysis.pl <query_filename> <database_location> <blast_program> <e-
value_cutoff> <output_file> <job_key>";
my $query_file = $ARGV[0] or die("Usage: $usage $!");
my $database = $ARGV[1] or die("Usage: $usage $!");
my $blast_program = $ARGV[2] or die("Usage: $usage $!");
my $cutoff = $ARGV[3] or die("Usage: $usage $!");
open(my $RESULTS, ">", $ARGV[4]) or die("Usage: $usage $!");
my $job_key = $ARGV[5] or die("Usage: $usage $!");

# Global variables
my $blaster;
my %organism_accessions;
my @organisms;
my $query_count = 0;
my $queries_with_hits = 0;
my $query_sequences;
my $start_time = time();


#----- Implementation -----#
#----- Implementation -----#
#----- Implementation -----#


# Create input stream, local BLAST instance if query file exists
if(-e $query_file) {
        $query_sequences = Bio::SeqIO->new(-file => $query_file, -format => 'Fasta');
        $blaster = Bio::Tools::Run::StandAloneBlast->new('program' => $blast_program, 'database'
=> $database, 'expect' => $cutoff);
} else {
        die("ERROR: query file '$query_file' does not exist");
}

# Load organism data
load_organisms();

# BLAST each gene query, process the result
while(my $query_sequence = $query_sequences->next_seq()) {

        debug("");
        debug("BLASTing gene '". $query_sequence->display_id() ."'...", 0);
        my $blast_results = $blaster->blastall($query_sequence);
        debug("Done!");
        $query_count++;
```

```
        debug("Processing BLAST result...", 0);
        my $gene_report = process_blast_result($blast_results->next_result());
        debug("Done!");

        if($gene_report ne "0") {
                print $RESULTS "$gene_report \n";
                $queries_with_hits++;
        }
}

if($queries_with_hits == 0) {
        print $RESULTS "The query genes have no bacterial homologs.";
}

close($RESULTS);
my $elapsed_time = time() - $start_time;
debug("Running time: $elapsed_time seconds");

my $dbi = DBI->connect("dbi:mysql:database=standage;host=psoda4.cs.byu.edu", "phylo", "", {
RaiseError => 1, AutoCommit => 1}) or die("ERROR: unable to connect to SQL database $!");
my $sql = "UPDATE queue SET job_status = 'complete' WHERE job_key = '$job_key' LIMIT 1;";
my $query = $dbi->prepare($sql) or die("ERROR: unable to prepare SQL $!");
$query->execute() or die("ERROR: unable to execute SQL query $!");
debug("[SQL] ". $sql);
debug("[SQLERR]". $DBI::errstr);


#----- Support methods -----#
#----- Support methods -----#
#----- Support methods -----#


# Print debugging statement
sub debug {
        my $line = shift();
        print "$line";
        if(scalar(@_) != 1) {
                print "\n";
        }
}

# Get an empty profile
sub get_empty_profile {
        my %profile;

        foreach(@organisms) {
                $profile{$_} = 0;
        }

        return %profile;
}

# Get the organism name from a given accession
sub get_organism_from_accession {
        my $accession = shift();
        foreach(@organisms) {
                my $org_accession = $organism_accessions{$_};
                if($accession =~ m/$org_accession/) {
                        return $_;
                }
        }

        print "WARNING: unable to match accession '$accession' to an organism in the database\n";
        return "fail";
#       die("ERROR: unable to match accession '$accession' to an organism in the database $!");
}

# Grab the name of each organism represented in the database, along with its associated
accessions
sub load_organisms {
```

```
	open(my $ORG, "<", "setup/genomes_with_accessions.txt") or die("ERROR: unable to find
organism file");
	while(<$ORG>) {
		chomp();
		my @line_values = split(/\t/);
		push(@organisms, $line_values[0]);
		$organism_accessions{$line_values[0]} = $line_values[1];
	}
	close($ORG);
}

# Create a phylogenetic profile given the blast result
sub process_blast_result {
	my $result = shift();
	my $hit_count = 0;
	my $gene_score = 0;
	my $query_accession = $result->query_name();
	my $output = "";
	my %profile = get_empty_profile();

	while(my $hit = $result->next_hit()) {
		$hit_count++;
		my $hit_organism = get_organism_from_accession($hit->accession());
		my $profile_test = $profile{$hit_organism};
		if($profile{$hit_organism} != 1) {
			$gene_score++;
		}
		if($hit_organism ne "fail") {
			$profile{$hit_organism} = 1;
		}
#my $hsp = $hit->next_hsp();
#my $hseq = $hsp->seq('hit');
#my $did = $hseq->annotation();
#print $hit->accession() ." ==> ". $did ."\n";
		my $profile_test = $profile{$hit_organism};
		if($profile{$hit_organism} != 1) {
			$gene_score++;
		}
		$profile{$hit_organism} = 1;
	}

	if($gene_score == 0) {
		return "0";
	}

	$output .= $result->query_name() ."\t". $gene_score ."\t". $hit_count ."\t";
	foreach(@organisms) {
		$output .= $profile{$_};
	}

	return $output;
}
```