Web-based SimpleAnalysis HOWTO

Richard Adams

Dept. Medical Genetics, University of Edinburgh

`<radams_at_staffmail.ed.ac.uk>`

2003-11-14

Revision History

Revision 0.1	2003-11-14	RA
First version		

This HOWTO tries to teach you to run a web based sequence analysis program using basic SimpleAnalysis rules.

Table of Contents

1. Introduction

There are several situations where it would be useful to run a web based sequence analysis program via a Perl script rather than using the web interface manually or downloading the program. Firstly, the source code or binaries may be unavailable or unavailable on your platform. Secondly, the analysis may depend on a regularly updated or large database that you don't wish to store or keep updated yourself. Thirdly, novel analyses are frequently available via a web server before being available for download.

The aim of the Bio::Tools::Analysis modules is to allow automatic submission of sequences to prediction servers to facilitate sequence annotation for low to medium numbers of sequences (perhaps tens to hundreds). The modules both submit the sequences and parse the results into a number of useful formats, usually including Bio::SeqFeature objects, Bio::Seq::MetaI sequence objects and a basic Perl data structure, as well as the raw output text.

At present the following prediction servers are supported, mainly reflecting my own research interests. Under current development are modules wrapping remote analyses using HMMER, Pfam, the ELM peptide motif database and SIFT for prediction of untolerated non-synonymous SNPs.

Analysis	Purpose	Reference
Domcut	Protein domain boundaries	Bioinformatics 19, 673-674 (2003)
NetPhos	Protein phosphorylation sites	J Mol Biol 294, 1351-1362 (1999)

Analysis	Purpose	Reference
GOR4	Protein Secondary structure	Meths. Enzymology 266, 540-553 (1996)
HNN	Protein Secondary structure	Bioinformatics 15,413-421 (1999)
Sopma	Protein Secondary structure	Comput Appl Biosci 11, 681-684 (1995)
Mitoprot	Mitochondrial cleavage site prediction	Eur J Biochem 241, 779-786 (1996)
ESEfinder	Exonic splice site enhancers	NAR 31, 3568-3571 (2003)

Table 1. Supported analyses

2. Simple Examples

The script below runs multiple analyses on a single sequence and parses the results into standard BioPerl sequence feature objects (Bio::SeqFeature::Generic objects to be precise).

```
# load up the modules

use Bio::Tools::Analysis::Protein::HNN;
use Bio::Tools::Analysis::Protein::Domcut;
use Bio::Tools::Analysis::Protein::MitoProt;
use Bio::Tools::Analysis::Protein::NetPhos;


our @ISA = qw(Bio::Tools::Analysis::SimpleAnalysisBase);

my $seq;# a Bio::Seq object
for my $method ( qw(Domcut MitoProt NetPhos HNN)) {

   #analyses need a Bio::PrimarySeq, not a Bio::Seq;
   my $tool = Bio::Tools::Analysis::Protein::$method->new(
        -seq => $seq->primary_seq);
   $tool->run();
   my @fts = $tool->result('Bio::SeqFeatureI');
   $seq->add_SeqFeature(@fts);
}
```

The above script runs several analyses using default parameters. All analyses have such defaults and in general only a sequence of the appropriate type is needed for the analysis to be submitted. A sequence can be added either in the constructor, as shown above, or by the seq() method.

```
my $primary_seq = new Bio::PrimarySeq(-seq=>$seq_as_string);
my $tool =  new Bio::Tools::Analysis::Protein::NetPhos();
$tool->seq($primary_seq);
```

Note that the only valid sequence format is a Bio::PrimarySeq object. This is in order to support multiple methods of retrieving the results. If you initially have a Bio::Seq object or Bio::RichSeq object (e.g., from a GenBank file) you can call its primary_seq() method to obtain a Bio::PrimarySeq object.

3. Retrieving results

If the run() method executes successfully, the raw output (stripped of HTML) is now stored in the Analysis object. All modules should return the raw report by a call to result() with no parameters. e.g.,

```
my $raw_report = $tool->result();
```

A second way is to retrieve a ready-parsed data structure:

```
my $parsed_report = $tool->result('parsed');
```

The data structure returned is described in the $RESULT_SPEC->{'raw'} hash reference and should always be a native Perl data structure.

A third way is to retrieve an array of sequence features:

```
my @fts = $tool->result('Bio::SeqFeatureI');
$seq->add_SeqFeature(@fts); # add features to sequence.
```

These are Bio::SequenceFeature::Generic features. Sometimes a module might use some code to judge the significance of a result prior to listing it as a feature, for example in the secondary structure prediction modules. The rules governing this are described in individual modules. For example, I have put in a rule that only runs of a minimum of 4 consecutive residues predicted to be beta sheet or alpha helix can be annotated as features - it makes no sense for a single isolated residue to be annotated as being in a helix. However you might want to filter the raw results yourself, in which case retrieving the results as Bio::Seq::Meta::Array type objects might be better.

4. Metasequences

Many analyses produce a list of scores, one for each residue in a sequence. For example, the protein secondary structure prediction program Sopma returns a list of probabilities for each residue being in helix, sheet, turn or coil. These modules make this data available in the form of meta sequences. These are sequence objects derived from Bio::PrimarySeq that have arrays of sequence associated data contained in the object. The meta sequence names should be listed in the individual module documentation. To retrieve results like this supply the string 'meta' as a parameter to the result() method. Meta sequence objects can access all the PrimarySeq object methods for accessing and manipulating the protein/DNA sequence and also have specific methods for accessing the result data.

```
$meta_seq = $analysis->result('meta');
```

This returns a sequence object with the raw analysis data accessible through methods e.g.,

```
my @scores1_20 = $meta_seq->named_sub_meta('Sopma_turn', 1,20);
```

returns an array of scores for the first 20 amino acids

```
my @allscores = $meta_seq->named_meta('Sopma_turn');
```

returns an array of scores for the whole sequence. The names of individual meta sequence names are listed in the module documentation.

Although a full description of how metasequence objects work isn't really the purpose of this HOWTO there is excellent documentation in Bio::Seq::MetaI and Bio::Seq::Meta::Array modules.

5. How to run the same analysis with varying parameters

You might want to run some analyses with varying parameters in order to determine the effect on the prediction. At present only the Sopma module takes alternative parameters i.e. arguments other than just the sequence. Any parameter that is settable on the web form should have a method of the same name to get/set its values, or alternatively it can be set in the constructor.

```
my $sopma = Bio::Tools::Analysis::Protein::Sopma->new();
$sopma->seq(seqobj->primary_seq);
$sopma->window_width(21);
```

So, let's suppose we want to investigate how varying the window_width parameter affects the secondary structure prediction for a sequence. We can do this inthe following manner:

```
my $seq = Bio::Seq->new(-seq => 'ASFATFDATFATFSTFATSFATFSTAF');
my $tool = Bio::Tools::Analysis::Protein::Sopma->new
  (-seq=>$seq->primary_seq);

for my $w_size(qw(11 13 15 17 19 21 23)) {

    $tool->window_width($w_size);  #set new parameter
    $tool->run(); #default parameters

    #2nd param is appended to metasequence name
    $tool->result('meta', $w_size);

    ## add to sequence features
    $seq->add_SeqFeature(
    $tool->result('Bio::SeqFeatureI'));
    $tool->clear(); #removes raw data from the previous analysis
}
# meta seq now has 28 arrays of metadats - 7 analyses and
# 4 prediction categories,

## now get meta_sequence
my $meta_seq = $tool->seq();
```

Only 3 new points are raised by this program. Firstly, each time the analysis is run it needs to be parsed immediately. The raw report data is removed from one analysis to another to reduce memory usage. Secondly, in order to distinguish the meta data each analysis needs to be given a specific identifier which is appended to the default name. So in this case the meta data names would be Sopma_meta|11, Sopma_meta|13, etc. Thirdly, the clear() method should be called between analyses of the same sequence to ensure the internal data fields that hold the raw report are removed. So if you want to keep the raw reports you need to store them after each analysis is returned.

So, how are features obtained from multiple runs distinguished? This information is contained in tags with the same name as the parameters, when the settings aren't the default ones. In other words, the features retain knowledge of the analysis method's parameters.

```
## carrying on from previous listing.... ##
##
## get all secondary structure features
my @sec_fts = grep{$_->primary_tag eq '2ary'}
                  $seq->all_SeqFeatures;
my @ww11 = grep{($_->each_tag_value('method'))[0]  eq 'Sopma' &&
                ($_->each_tag_value('window_width'))[0] == 11}
                @sec_fts;

## now onto comparison..... ##
```

6. Caveats

The problem with these analyses is their speed of execution, which obviously depends on your network speed, the complexity of the analysis and the speed of the host. Moreover, it is usually polite to leave a second or two between requests to avoid blocking the server, which means that to annotate a single protein sequence with all of the X methods available may take over a minute. Certainly these modules are unsuitable for genome scale analysis and are designed for use with smaller numbers of sequences. However, the code in the result() methods should still be usable for parsing analyses run locally when the local output and webpage output are the same format.

7. Interested in developing your own Analysis module?

Most of the hard work is done by Bio::WebAgent and Bio::SimpleAnalysisBase. Any module must inherit from this latter module as it contains methods common to all analyses.

For a minimal prediction server which takes just sequence as a parameter (e.g., Domcut), only 3 methods need to be explicitly written.

1. _run() which mimics the web form, submits the sequence and extracts the raw text data of the result from the HTML.

2. result() which parses the raw data into whatever useful format you wish. Usually these include SeqFeature objects, standard Perl data structures and meta sequences if there is a result for each residue in the sequence.

3. _init() which imports the analysis specifications into the Analysis object

As well as these methods, various file scoped lexical hashes need to be defined which hold constant data about the analysis, and the analysis result. These are useful for reference and are incorporated into the analysis objects.

For more complicated prediction programs with analysis specific parameters (e.g., Sopma), get/set methods need to be written. Also, if any of these parameters need special error checking then a check_parameters() needs to be written as well. The nature of these parameters should be listed in a hash referenced by the $INPUT_SPEC variable.

Alternatively, mail me with your suggestion and I'll try to put one together. It is my intent to keep the modules up to date with new analysis programs as they become available which are not included in the usual protein annotations and will be glad to hear of new suggestions.

8. Acknowledgments

These modules depend on several recently developed modules and wouldn't work at all without them: Bio::Seq::Meta modules by Chad Matsalla, Aaron Mackey and Heikki Lehvaslaiho, Bio::WebAgent by Heikki Lehvaslaiho, and Bio::SimpleAnalysisI by Martin Senger.