Bio::Graphics HOWTO

Lincoln Stein, *Cold Spring Harbor Laboratory* [http://www.cshl.org]
`<lstein@cshl.org>`

2002-09-01

Revision History

Revision 0.2	2003-05-15	lds
	Current as of BioPerl 1.2.2	

This HOWTO describes how to render sequence data graphically in a horizontal map. It applies to a variety of situations ranging from rendering the feature table of a GenBank entry, to graphing the positions and scores of a BLAST search, to rendering a clone map. It describes the programmatic interface to the Bio::Graphics module, and discusses how to create dynamic web pages using Bio::DB::GFF and the gbrowse package.

Table of Contents

2. Preliminaries

Example 1. Rendering the simple blast hit file (render_blast1.pl)

```perl
#!/usr/bin/perl

# This is code example 1 in the Graphics-HOWTO
use strict;
use Bio::Graphics;
use Bio::SeqFeature::Generic;

my $panel = Bio::Graphics::Panel->new(-length => 1000,-width  => 800);
my $track = $panel->add_track(-glyph => 'generic',-label  => 1);

while (<>) { # read blast file
  chomp;
  next if /^\#/;  # ignore comments
  my($name,$score,$start,$end) = split /\t+/;
```

```
% render_blast1.pl data1.txt | display -
```

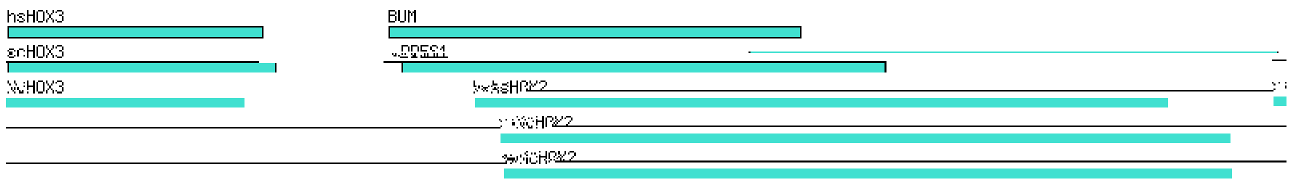

Figure 2. Rendering BLAST hits

Users of operating systems that don't support pipes can simply redirect the output to a file and view it in their favorite image program.

4. Adding a Scale to the Image

This is all very nice, but it's missing two essential components:

- It doesn't have a scale.
- It doesn't distinguish between hits with different scores.

Example 2 fixes these problems

eeeeeee

Example 2. Rendering the blast hit file with scores and scale

```perl
#!/usr/bin/perl

# This is code example 2 in the Graphics-HOWTO
use strict;
use lib '/home/lstein/projects/bioperl-live';
use Bio::Graphics;
use Bio::SeqFeature::Generic;

my $panel = Bio::Graphics::Panel->new(-length => 1000,
                                      -width  => 800,
                                      -pad_left => 10,
                                      -pad_right => 10,
                                     );
my $full_length = Bio::SeqFeature::Generic->new(-start=>1,-end=>1000);
$panel->add_track($full_length,
                  -glyph   => 'arrow',
                  -tick    => 2,
                  -fgcolor => 'black',
                  -double  => 1,
                 );

my $track = $panel->add_track(-glyph => 'graded_segments',
                              -label  => 1,
                              -bgcolor => 'blue',
                              -min_score => 0,
```

In lines 18-22, we get a bit fancier with the blast hit track. Now, instead of creating a generic glyph, we use the "graded_segments" glyph. This glyph takes the specified background color for the feature, and either darkens or lightens it according to its score. We specify the base background color (-bgcolor => 'blue'), and the minimum and

Example 3. Rendering the blast hit file with scores and scale

```
#!/usr/bin/perl

# This is code example 3 in the Graphics-HOWTO
use strict;
use lib '/home/lstein/projects/bioperl-live';
use Bio::Graphics;
use Bio::SeqFeature::Generic;

my $panel = Bio::Graphics::Panel->new(-length => 1000,
```

Example 4. Parsing and Rendering a Real BLAST File with Bio::SearchIO

```
#!/usr/bin/perl
```

```
    $feature->add_sub_SeqFeature($hsp,'EXPAND');
  }

  $track->add_feature($feature);
}

print $panel->png;
```

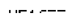

Figure 5. Output from the BLAST parsing and rendering script

The next section will demonstrate how to parse and display feature tables from GenBank and EMBL.

Important

Remember that if you are on a Windows platform, you need to put STDOUT into binary mode so that the PNG file does not go through Windo

Example 5.

```
my @notes;
foreach (qw(product gene)) {
  next unless $feature->has_tag($_);
  @notes = $feature->each_tag_value($_);
```