Introduction to Perl & Bioperl(II) Perl Module and Bioperl

Xuefeng Zhao
L. H. Baker Center, ISU

Reference

1. Chapter 12 in Bioinformatics by David W. Mount

2. www.bioperl.org and links on the same web page:

- **BioPerl 1.4 Tutorial**
- **Pasteur Institute Bioperl Course**
- **BioPerl 1.4 Module Documentation**

Objectives

- Create simple Perl modules
- Using Bioperl modules

Outline

Perl/BioPerl Modules

- Why Perl Modules?
- Where to download Perl Modules?
- Where to download BioPerl Modules?
- How to use modules in your script?
- Create and use your own Perl Modules
- Compare Bio::Seq module with your mSeqObj module

Why Perl Modules?

--Reuse the code
--Easy to manage codes
--separate codes to avoid namespaces to collide

Where to Download?

Comprehensive Perl Archive Network:
http://www.cpan.org/

How to install to your local Windows?

DOS cmd: perl –e shell –MCPAN
cpan> install your-module-name

What packages are installed on your computer?

ppm>query *

How to use the modules in your script

```
#Example: get the sequence from GeneBank

use strict;            # use strict pragma to ensure all variables are safe

# Libraries and modules are in the sub-directory Lib. In my laptop,
# BioPerl modules are installed in C:\Perl\site\lib\Bio\.
# The full name of GenBank.pm is C:\Perl\site\lib\Bio\DB\GenBank.pm
# To load the module , use the compiler directive "use".
# The modules in these subdirectories are loaded using the subdirectory name, two
    colons, and the module name

use Bio::Seq;
use Bio::DB::GenBank;

# Create a new object of GeneBank using the new method
# The -> operator is used to call a method associated with a class.

my $gb = Bio::DB::GenBank->new();

# Call the method get_Seq_by_id to get the sequence of   MUSIGHBA1,
# the return is a Seq object.
my $seq1 = $gb->get_Seq_by_id('MUSIGHBA1');

print "Sequence is:\n";
print  $seq1->seq, "\n";
```

To verify the sequence

1. Go to GeneBank: http://www.ncbi.nlm.nih.gov/Genbank/

2. Search the database of nucleotide, enter the Gene ID:MUSIGHBA1

3. Compare the output from Genebank web site and from you perl script.

Create your own Hello Perl moudule

```
package HelloAgain;
# The first statement is package declaration. the package name must match the file
# name. The file name ends with '.pm'. our package name is HelloAgain.pm

use strict;
our $VERSION= "2.0";
# our declares the listed variables to be valid global ones within the enclosing block
# and file. it has the same scoping rules as a "my" declaration,
# but does not create a local variable. it can be accessed by
# $HelloAgain::VERSION, but $msg can not

my $msg="Hello, BCB \n";   #scope: to the whole package, include sub

sub hello_msg {
 return $msg;
}

1;
# the package must return a true value so that Perl knows it was loaded ok when it
# is used. it can be any true values, but 1 is always TRUE.
```

Use your own Hello Perl moudule

```
# use lib ' Path/to/the/module';
# if you installed by yourself, and not in the same directory with your perl scripts
# use moulde_name to load a module.

use strict;
use  HelloAgain;
my $str= HelloAgain::hello_msg();
print $str, "\n";
print "Version: ", $HelloAgain::VERSION, "\n";
```

Note: the hello.pl should be in the same directory as that of HelloAgain.pm so that you do not need to INSTALL your module.

Create your own SeqObj Perl moudule(Outline)

```
package mSeqObj;  # The first statement is package declaration.
use strict;
our $VERSION= "2.0";

my $DNAstr="ATGCTTTT";   #scope: to the whole package, include sub
my ($num_A, $num_T, $num_C, $num_G, $num_error)=(0,0,0,0,0);

our $o_file_name="";         # the output file name can be assigned by the caller
sub GC_counter()
{            statements;
             return @counter_arr;
}
sub out2File_FASTA()
{
   open (OUT, ">$o_file_name");
   print OUT "\>DNA-ID\|no comment\n";
   print OUT $DNAstr,"\n";
   close(OUT);
}
1;
```

Use your own SeqObj moudule

```
use strict;
use  mSeqObj;
my @counters = mSeqObj::GC_counter();

$mSeqObj::o_file_name="mydnaseq.fa";

mSeqObj::out2File_FASTA();

print "Version: ", $mSeqObj::VERSION, "\n";
print @counters, "\n";
```

Note: the caller script and module script should be in the same directory so that you do not need to INSTALL your module.

FASTA format

http://ngfnblast.gbf.de/docs/fasta.html

Where to Download BioPerl modules?

www.bioperl.org

Links on this Page:

BioPerl 1.4 Tutorial
Pasteur Institute Bioperl Course
BioPerl 1.4 Module Documentation

Compare your SeqObj with Bio::Seq

1. Read the documentation online

2. To use the module:

```
#!/bin/perl -w
 use strict;

 use Bio::Seq;

 my $seq_obj = Bio::Seq->new(-seq => "aaaatgggggggggggccccgtt",
                             -alphabet => 'dna' );

print $seq_obj->seq;
```

BioPerl: Object Oriented Perl

- Objects are module-specific references to data
- A module can describe multiple objects
  - Bio::SeqIO::fasta
  - Bio::SeqIO::GenBank
- **->** send information about the data

Examples of creating an object and performing methods on it:

- **$seqs = Bio::SeqIO->new(-file => "$inFile",**
  '-format' => 'Fasta'); # makes a SeqIO object
- **$seqobj = $seqs->next_seq(); # makes a Seq object**
- **$rawseq = $seqobj->seq();**
- **$rev_comp = $seqobj->revcom->seq();**

Summary

Create your own Perl modules

Call BioPerl Modules to get data from GeneBank

Practice

***Question on script example 1*---BioPerl: Read sequences from a GeneBank format file and write them to a FASTA format file (p. 579)**

***Question on script example 2*---BioPerl: Using the GLOB function to obtain a list of files with a specified extension (p. 580)**

***Question on script example 5*---Perl: Reading files and looking for text patterns (p. 595)**

***Question on script example 7*---Perl: Pattern Substitution an incremental development of scripts (p. 600)**