分子生物计算
(*Perl 语言编程*)

天津医科大学
生物医学工程与技术学院

2019-2020 学年上学期（秋）
2017 级生信班

第 10..13 章　GenBank、PDB、BLAST、其他

伊现富（Yi Xianfu）

天津医科大学（TIJMU）
生物医学工程与技术学院

2019 年 11 月

教学提纲

1. 模式匹配
2. 输入记录分隔符
3. 读取文件
4. 文件夹处理
5. 格式化输出
6. 与外部程序进行交互
7. 浮点数比较
8. 引用
9. 模块
10. BioPerl

教学提纲

1. 模式匹配
2. 输入记录分隔符
3. 读取文件
4. 文件夹处理
5. 格式化输出
6. 与外部程序进行交互
7. 浮点数比较
8. 引用
9. 模块
10. BioPerl

```perl
my $name = "manfred";

if ($name =~/fred/) {
    print "You could be fred\n";
}
#You could be fred

if ($name =~ /\bfred\b/) {
    print "You ARE fred\n";
}
```

模式匹配 | 界定符 | =~m

```
1 if ( $line =~ /^\/\/\n/ ) {
2   last;
3 }
4
5 if ( $line =~ m!//\n! ) {
6   last;
7 }
```

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

use warnings;

"AAC\nGTT" =~ /^.*$/;
print "Without /m:\n", $&, "\n";
#Without /m:
#Use of uninitialized value $& in print at
  XXX.pl line N.

"AAC\nGTT" =~ /^.*$/m;
print "With /m:\n", $&, "\n";
#With /m:
#AAC
```

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

use warnings;

"AAC\nGTT" =~ /^.*$/;
print "Without /s:\n", $&, "\n";
#Without /s:
#Use of uninitialized value $& in print at
  XXX.pl line N.

"AAC\nGTT" =~ /^.*$/s;
print "With /s:\n", $&, "\n";
#With /s:
#AAC
#GTT
```

模式匹配 | 捕获

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

use strict; use warnings;

my $alphabet = join "", 'a' .. 'z';
$alphabet =~ /k(lmnop)q/;
print $1, "\n\n";
#lmnop

$alphabet =~ /(((a)b)c)/;
print "First pattern = ",  $1, "\n";
print "Second pattern = ", $2, "\n";
print "Third pattern = ",  $3, "\n";
#First pattern = abc
#Second pattern = ab
#Third pattern = a
```

```perl
my $string = "File code=123 name=test.txt";
if ($string =~ /code=(\d+)\s+name=([\w\.]+)/) {
    print "Code is $1\nName is '$2'\n";
}
```

```perl
my %q_count;
while (<>) {
    while (/\b(Q\w+)\b/g) {
        ++$q_count{$1};
    }
}
foreach my $word (sort {$q_count{$b}<=>$q_count{$a}} keys %q_count) {
    print "The word $word appeared ",$q_count{$word}," times\n";
}
```

When not to use Regular Expressions

- Splitting delimited data: `split`
- Swapping single characters: `tr`
- Extracting fixed position data: `substr`
- Finding the position of an exact string: `index`

模式匹配 | index

```
1 my $string = "xxxxxxhixxxxxxxxxxhixxxxxxxxxhi";
2 my $lastpos = 0;
3
4 while (1){
5     my $pos = index($string,"hi",$lastpos);
6     last if ($pos == -1); # Substring not
    found
7     print "Found hi at index $pos\n";
8     $lastpos = ++$pos;
9 }
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

输入记录分隔符

```
1 my $save_input_separator = $/;
2
3 $/ = "//\n";
4 $record = <GBFILE>;
5
6 $/ = $save_input_separator;
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

读取文件 | tell & seek

```
1 for (;;) {
2     for ($curpos = tell(FILE); $_ = <FILE>;
  $curpos = tell(FILE)) {
3         # search for some stuff and put it
  into files
4     }
5     sleep($for_a_while);
6     seek(FILE, $curpos, 0);
7 }
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

文件夹处理 | 递归

```perl
#!/usr/bin/perl
use strict; use warnings;
list_recursively('pdb');

sub list_recursively {
    my ($directory) = @_;
    my @files = ();
    unless ( opendir( DIRECTORY, $directory ) ) {
        print "Cannot open directory $directory!\n";
        exit;
    }
    @files = grep ( !/^\.\.?$/, readdir(DIRECTORY) );
    closedir(DIRECTORY);
    foreach my $file (@files) {
        if ( -f "$directory/$file" ) {
            print "$directory/$file\n";
        }
        elsif ( -d "$directory/$file" ) {
            list_recursively("$directory/$file");
        }
    }
}
```

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

use strict;
use warnings;
use File::Find;
#perldoc File::Find

find( \&my_sub, ('pdb') );

sub my_sub {
    -f and ( print $File::Find::name, "\n" );
}
```

```
1  # <*>
2  my @files = <*.doc>;
3  print "I have ",scalar @files," doc files in
     my work directory\n";
4
5  # glob
6  my @files2 = glob("*.rtf");
7  print "I have ",scalar @files2," rtf files in
       my work directory\n";
8
9  chdir ("/home") or die "Can't move to work
     directory: $!";
10 while (my $file = <*.doc>) {
11     print "Found file $file\n";
12 }
```

```perl
#!/usr/bin/perl
use warnings; use strict;

# If you want to open a file in a location
  relative to the location of your script
  then you can use the FindBin module to get
  the filesystem location of your Perl
  program.
use FindBin qw($Bin);

# This code will add the ExtraModules
  directory to the module search path.
use lib "$Bin/ExtraModules";

print "Script is located in $Bin\n";
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

格式化输出 | printf

```
1  while(<>) {
2    /^ATOM/ or next;
3
4    my($n, $x, $y, $z, $element)
5      = ($_ =~ /^.{6}(.{5}).{19}(.{8})(.{8})
     (.{8}).{22}(..)/);
6
7    $n       =~ s/^\s*//;
8    $element =~ s/^\s*//;
9
10   if (($n == 1) or ($n == 1078)) {
11     printf "%8.3f%8.3f%8.3f %2s\n", $x, $y,
     $z, $element;
12   }
13 }
```

格式化输出 | printf

```
1 my $first  = '3.14159265';
2 my $second  = 76;
3 my $third = "Hello world!";
4
5 printf STDOUT "A float: %6.4f An integer: %-5
    d and a string: %s\n", $first, $second,
    $third;
6 #A float:  3.1416 An integer: 76    and a
    string: Hello world!
```

格式化输出 | here 文档

```
1  #!/usr/bin/perl
2  use strict; use warnings;
3  my $DNA = 'AAACCCCCCGGGGGGGGTTTTTT';
4  for ( my $i = 0 ; $i < 2 ; ++$i ) {
5      print <<HEREDOC;
6       On iteration $i of the loop!
7      $DNA
8
9  HEREDOC
10 }
11 #      On iteration 0 of the loop!
12 #     AAACCCCCCGGGGGGGGTTTTTT
13 #
14 #      On iteration 1 of the loop!
15 #     AAACCCCCCGGGGGGGGTTTTTT
16 #
```

格式化输出 | format & write

```
1  #!/usr/bin/perl
2  use strict; use warnings;
3  my $id          = 'A0000';
4  my $description = 'Highly weird DNA.  This DNA is so
     unlikely!';
5  my $DNA = '
     AAAAAACCCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGGTTTTTTTTTTTTTTTTTT
     ';
6  # Define the format
7  format STDOUT =
8  # The header line
9  >@<<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<...
10 $id,         $description
11 # The DNA lines
12 ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<~~
13 $DNA
14 .
15 # Print the fasta-formatted DNA output
16 write;
```

格式化输出 | format & write

```
1 >A0000       Highly weird DNA.  This DNA is so un...
2 AAAAAACCCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGTTTTTTTT
3 TTTTTTTTTTTT
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

用途

Using the functions described in this next slide it is straightforward to either pass data through an external program as part of your Perl script, use Perl as a glue language to automate the execution of other programs, or simply use Perl as a convenient way to launch anther program.

三种方法

There are three different functions you can use within Perl to launch an external program. These are `system`, backticks (`` ` ` ``) and `exec` and they all have slightly different purposes.

用途

Using the functions described in this next slide it is straightforward to either pass data through an external program as part of your Perl script, use Perl as a glue language to automate the execution of other programs, or simply use Perl as a convenient way to launch anther program.

三种方法

There are three different functions you can use within Perl to launch an external program. These are `system`, backticks (` `) and `exec` and they all have slightly different purposes.

system

System is used where you want to launch an external program and check that it worked, but you don't need to collect any data back from it.

backticks

If you want to get hold of the output of programs then you need to use backticks rather than system.

exec

Exec is a somewhat unusual function in perl as it causes execution of your perl program to end immediately, and your running perl program is replaced by whatever program you specify.

system

System is used where you want to launch an external program and check that it worked, but you don't need to collect any data back from it.

backticks

If you want to get hold of the output of programs then you need to use backticks rather than system.

exec

Exec is a somewhat unusual function in perl as it causes execution of your perl program to end immediately, and your running perl program is replaced by whatever program you specify.

system

System is used where you want to launch an external program and check that it worked, but you don't need to collect any data back from it.

backticks

If you want to get hold of the output of programs then you need to use backticks rather than system.

exec

Exec is a somewhat unusual function in perl as it causes execution of your perl program to end immediately, and your running perl program is replaced by whatever program you specify.

```perl
my $filename = $ARGV[0];
my $stride = '/usr/local/bin/stride';
my $options = '';
# 捕获输出
my @results = `$stride $options $filename`;
my $now = `date`;
my @functions = qw( int rand length );
my %about;
foreach (@functions) {
   #$about{$_} = `perldoc -t -f $_`;
   $about{$_} = qx(perldoc -t -f $_);
}

# 不捕获输出，返回值为程序退出状态
system "$stride $options $filename";
system 'date';
system 'tar', 'cvf', $tarfile, @dirs;
```

Piping data out of an external program

```
1 open (my $log, "tail -f /var/log/httpd/
    access_log |") or die "Can't open pipe to
    web logs: $!";
2
3 while (<$log>) {
4     if (/Safari/) {
5         print "Oooh, a Mac user...\n";
6     }
7 }
```

Piping data into an external program

```
1 open (my $zip, "| zip compress.zip -") or die
    "Can't open pipe to zip: $!";
2
3 print $zip "I want to be smaller...";
4
5 close $zip or die "Can't close pipe to zip :
  $!";
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

浮点数比较

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

if ( 10 / 3 == ( ( 1 / 3 ) * 10 ) ) {
    print "Success!\n";
}
else { print "Failure!\n"; }
#Failure!

if ( abs( 10/3 - ( ( 1/3 ) * 10 ) ) < 1e-10 ) {
    print "Right!\n";
    print "E=", abs(10/3 - ( (1/3) * 10 )), "\n";
}
else { print "Wrong!\n"; }
#Right!
#E=4.44089209850063e-16
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

```perl
my @array = (1, 2, 3, 4);
my $slow_arrayref = \@array;
my $quick_arrayref = [1, 2, 3, 4];

my %hash = (
    dog => 'woof',
    cat => 'meow',
);
my $slow_hashref = \%hash;
my $quick_hashref = {
    dog => 'woof',
    cat => 'meow',
};
```

引用 | 解引用

```
1 my $arrayref = [10, 20, 30];
2 print "First element is ", $$arrayref[0],  "\
    n";
3 print "First element is ", $arrayref->[0], "\
    n";
4
5 my $hashref = {
6     duck => 'quack',
7 };
8 print "The duck says ", $$hashref{duck},  "\n
    ";
9 print "The duck says ", $hashref->{duck}, "\n
    ";
```

```
1  my @original = (2, 4, 6, 8);
2  my @copy = @original;
3  for (0..$#copy){ $copy[$_] *= 10; }
4  print "Copy says ",$copy[1]," but original
     was ",$original[1];
5  # The copy is altered so the second element
     is 40, but the original still says 4.
6
7  my $original = [2, 4, 6, 8];
8  my $copy = $original;
9  for (0..(@$copy-1)){ $copy->[$_] *= 10; }
10 print "Copy says ",$copy->[1]," but original
     was ",$original->[1];
11 # Both the original and copy references point
      to an array whose second element is 40.
```

Country	Population	Language	Currency
UK	60,441,457	English	Pounds
France	60,656,178	French	Euros
Ireland	4,015,676	English/Irish	Euros

```
1  my %countries;
2  my %uk_info = (
3      population => 60441457,
4      language => 'English',
5      currency => 'Pounds',
6  );
7  my %france_info = (
8      population => 60656178, language => 'French',
     currency => 'Euros',
9  );
10 my %ireland_info = ( ... );
11
12 $countries{uk} = \%uk_info;
13 $countries{france} = \%france_info;
14 #$countries{ireland} = \%ireland_info;
15
16 print "Population of France is ",
17     $countries{france}->{population},"\n";
```

```
my %countries = (
    uk => {
        population => 60441457,
        language => 'English',
        currency => 'Pounds',
    },
    france => {
        population => 60656178,
        language => 'French',
        currency => 'Euros',
    },
    ireland => {
        ...
    },
);

print "Population of France is ",
    $countries{france}->{population},"\n";
```

```
1 my %countries;
2
3 $countries{uk}      -> {population} = 60441457;
4 $countries{france} -> {population} = 60656178;
5 $countries{france} -> {currency}   = 'Euros';
6
7 print "Population of France is ",
8     $countries{france}->{population},"\n";
```

Data: country.txt

```
|----------+------------+---------------+-----------|
|  Country | Population | Language      | Currency  |
|----------+------------+---------------+-----------|
|  UK      | 60,441,457 | English       | Pounds    |
|  France  | 60,656,178 | French        | Euros     |
|  Ireland | 4,015,676  | English/Irish | Euros     |
|----------+------------+---------------+-----------|
```

Task

Q: Population of France

A: 60,656,178

Data: country.txt

```
|----------+------------+---------------+-----------|
|  Country | Population | Language      | Currency  |
|----------+------------+---------------+-----------|
|  UK      | 60,441,457 | English       | Pounds    |
|  France  | 60,656,178 | French        | Euros     |
|  Ireland | 4,015,676  | English/Irish | Euros     |
|----------+------------+---------------+-----------|
```

Task

Q: Population of France

A: 60,656,178

shell

```
1 grep -i -w France country.txt | cut -f2
```

csvkit

```
1 # Method 1
2 csvsql --query "select Population from
    country where Country=='France'" country.
    txt | csvlook
3 # Method 2
4 csvgrep -t -c Country -m France country.txt |
     csvcut -c Population | csvlook
```

shell

```
1 grep -i -w France country.txt | cut -f2
```

csvkit

```
1 # Method 1
2 csvsql --query "select Population from
    country where Country=='France'" country.
    txt | csvlook
3 # Method 2
4 csvgrep -t -c Country -m France country.txt |
     csvcut -c Population | csvlook
```

Perl-hash

```
#!/usr/bin/perl
use strict; use warnings;
my $fi = "country.txt";
my ( @header, @fields, %countries);
open my $IN, '<', $fi or die "$0 : failed to open  input file '
  $fi' : $!\n";
while (<$IN>) {
    chomp;
    if ( $. == 1 ) {
        @header = map { lc } split /\t/;
    }
    else {
        @fields = map { lc } split /\t/;
        for ( my $i = 1 ; $i < @header ; $i++ ) {
            $countries{$fields[0]}->{$header[$i]}=$fields[$i];
        }
    }
}
close $IN or warn "$0: failed to close input file '$fi': $!\n";
print "Population of France is ", $countries{france}->{
  population}, "\n";
```

Perl-Data::Table

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5  use Data::Table;
6
7  my $countries = Data::Table::fromFile("country.txt
     ");
8
9  print "Population of France is ",
10   $countries->
11   match_pattern_hash('$_{Country} eq "France"')->
12   col('Population'),
13   "\n";
```

R

```r
#!/usr/bin/Rscript

library(readr)
library(dplyr)

countries <- read_tsv("country.txt")
countries %>%
  filter(Country=="France") %>%
  select(Population) %>%
  pull()
```

```perl
my @experiments = (
    {
        sample_id => 1,
        sample_name => 'kidney',
        sample_measures => [12,56,34,65,76],
    },
    {
        sample_id => 4,
        sample_name => 'liver',
        sample_measures => [24,66,12,17,26],
    },
    { ... },
);

foreach my $expt (@experiments) {
    print "The first measure for sample ",
          $expt->{sample_id},
          " (",$expt->{sample_name},") was ",
          $expt->{sample_measures}->[0],"\n";
}
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

```
1  # METHOD 1: cpan
2  #Open cpan shell
3  sudo cpan #Or
4  perl -MCPAN -e shell
5  #Install modules
6  cpan>install Date::Calc
7
8  # METHOD 2: cpanm
9  #Install cpanm
10 #perlbrew is an admin-free perl installation
     management tool
11 perlbrew install-cpanm
12 #Install modules
13 cpanm Bio::Perl
14
15 # METHOD 3: manual
```

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 use Date::Calc qw(:all);
7 #use Date::Calc qw(Days_in_Year Days_in_Month
    );
8
9 print "In Feb 2020 there are ",Days_in_Month
    (2020,2)," days";
```

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

use warnings; use strict;

use LWP::UserAgent;

my $ua = LWP::UserAgent->new;
$ua->timeout(10);
$ua->env_proxy;

my $response=$ua->get('http://search.cpan.org/');

if ($response->is_success) {
    print $response->decoded_content;
}
else {
    die $response->status_line;
}
```

```
1  #!/usr/bin/perl
2
3  use warnings;
4  use strict;
5
6  package Example::Module;
7
8  # Module code goes here
9
10 1;
```

```perl
#!/usr/bin/perl
use warnings; use strict;

use Exporter;
our @ISA = qw(Exporter);
our @EXPORT_OK = qw(will_be_exported);

sub public {
    # This is designed to be seen, but can only be addressed by using its fully qualified name eg Example::Functional::public()
}
sub _private {
    # This is only for internal use and shouldn't be used from outside the module.
}
sub will_be_exported {
    # This can be exported into the namespace of the calling program
}

1;
```

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

use warnings;
use strict;

package Example::Object_oriented;

sub new {
    # This creates the refernece which is going to be our object
    my $hashref = {};
    # We then call bless to associate it with this module.
    bless $hashref;
    # Finally we return it so the calling program can start using it.
    return $hashref;
}
```

```
16 sub save_value {
17     # This subroutine takes a single arguement which it
   stores in the hash reference. The $object is provided
   automatically as the first argument to every object
   oriented subroutine (other than 'new').
18     my ($object, $new_value) = @_;
19     $object->{value} = $new_value;
20 }
21 sub get_value {
22     # This subroutine retrieves a value which was
   previously stored via the save_value subroutine. If
   there isn't a value to retrieve it returns the
   undefined value.
23     my ($object) = @_;
24     if (exists $object->{value}) {return $object->{value
   };}
25     else { return undef; }
26 }
27
28 1;
```

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

use warnings;
use strict;

use Example::Object_oriented;

my $object = Example::Object_oriented->new();

$object->save_value("Hello");

print "The object says '". $object->get_value
  () . "'\n";
```

教学提纲

1 模式匹配
2 输入记录分隔符
3 读取文件
4 文件夹处理
5 格式化输出
6 与外部程序进行交互
7 浮点数比较
8 引用
9 模块
10 BioPerl

BioPerl

```
1  use Bio::SeqIO;
2
3  my $string = ">SEQ1\nacgt\n>revseq1\ntgca ";
4  my $format = "fasta";
5
6  my $stringfh = IO::String->new($string);
7  open($stringfh, "<", $string) or die "Could not open
     string for reading: $!";
8
9  my $seqio = Bio::SeqIO-> new(-fh     => $stringfh,
10                               -format => $format,
11                              );
12
13 while( my $seq = $seqio->next_seq ) {
14    print $seq->id . ' = ' . $seq->seq() . "\n";
15 }
```

BioPerl

```perl
use strict; use Bio::SeqIO;
my $input_file = shift;
my $seq_in = Bio::SeqIO->new( -format => 'embl',
                              -file   => $input_file,
                            );
my @seq_array;
while ( my $seq = $seq_in->next_seq() ) {
   push(@seq_array,$seq);
}
@seq_array = sort { $a->length <=> $b->length }
  @seq_array;
my $total = 0;
my $count = 0;
for my $seq ( @seq_array ) {
   $total += $seq->length;
   $count++;
}
print "Mean length ", $total/$count, " Median ",
      $seq_array[$count/2]->length, "\n";
```

BioPerl

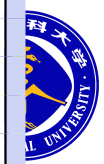
```perl
use Bio::SeqIO;

my $usage = "x2y.pl infile informat outfile outformat ";
my $infile = shift or die $usage;
my $informat = shift or die $usage;
my $outfile = shift or die $usage;
my $outformat = shift or die $usage;

my $seq_in = Bio::SeqIO->new( -file   => "$infile",
                              -format => $informat,
                            );
my $seq_out = Bio::SeqIO->new( -file   => ">$outfile",
                               -format => $outformat,
                             );
while (my $inseq = $seq_in->next_seq) {
    $seq_out->write_seq($inseq);
}
```

BioPerl

```
use Bio::SeqIO;
my $usage = "splitgb.pl infile "; my $infile = shift or die $usage;
my $inseq = Bio::SeqIO->new( -file   => "<$infile", -format => 'Genbank',);
my %outfiles = ( human => {
                          Genbank => Bio::SeqIO->new(
                                                     -file   => '>human.gb',
                                                     -format => 'Genbank',
                                                     ),
                          Fasta   => Bio::SeqIO->new(
                                                     -file   => '>human.fa',
                                                     -format => 'Fasta',
                                                     ),
                          },
                 other => {
                          Genbank => Bio::SeqIO->new(
                                                     -file   => '>other.gb',
                                                     -format => 'Genbank',),
                          Fasta   => Bio::SeqIO->new(
                                                     -file => '>other.fa',
                                                     -format => 'Fasta',),
                          }
                );
while (my $seqin = $inseq->next_seq) {
   if ($seqin->species->binomial =~ m/Homo sapiens/) {
       $outfiles{'human'}->{'Genbank'}->write_seq($seqin);
       $outfiles{'human'}->{'Fasta'}->write_seq($seqin);
   } else {
       $outfiles{'other'}->{'Genbank'}->write_seq($seqin);
       $outfiles{'other'}->{'Fasta'}->write_seq($seqin);
   }
}
```



BioPerl

```perl
use Bio::DB::GenBank;
use Bio::DB::Query::GenBank;

$query = "Arabidopsis[ORGN] AND topoisomerase[TITL] and
  0:3000[SLEN]";
$query_obj = Bio::DB::Query::GenBank->new(-db => '
  nucleotide',
                                          -query =>
  $query );

$gb_obj = Bio::DB::GenBank->new;

$stream_obj = $gb_obj->get_Stream_by_query($query_obj);

while ($seq_obj = $stream_obj->next_seq) {
    # do something with the sequence object
    print $seq_obj->display_id, "\t", $seq_obj->length,
  "\n";
}
```

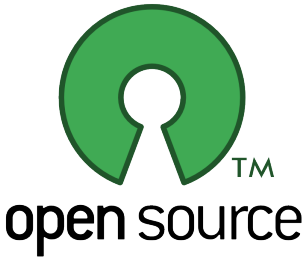
BioPerl

```perl
#!/usr/bin/perl
use strict; use warnings;
use Bio::DB::Query::GenBank; use Bio::DB::GenBank;
use Bio::SeqIO;

my $query_string = $ARGV[0];
my $fo_fa=$query_string.".fa"; my $fo_gb=$query_string.".gb";
my $query = Bio::DB::Query::GenBank->new(
    -db => 'nucleotide', -query => $query_string
);
my $gb = Bio::DB::GenBank->new;
my $stream = $gb->get_Stream_by_query($query);

my %outfiles = (
    Fasta => Bio::SeqIO->new( -file => ">$fo_fa", -format => '
  Fasta',),
    Genbank => Bio::SeqIO->new( -file => ">$fo_gb", -format => '
  Genbank',),
);
while ( my $seq = $stream->next_seq ) {
    $outfiles{'Fasta'}->write_seq($seq);
    $outfiles{'Genbank'}->write_seq($seq);
}
```

Powered by



TEX LATEX XETEX Beamer