

# 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

2016-2017 学年上学期 (秋)  
2014 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

2016 年 12 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl

- 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl



```
1 my $name = "manfred";
2
3 if ($name =~ /fred/) {
4     print "You could be fred\n";
5 }
6 #You could be fred
7
8 if ($name =~ /\bfred\b/) {
9     print "You ARE fred\n";
10 }
```



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



# 模式匹配 | 修饰符 | /m

```
1 #!/usr/bin/perl
2
3 use warnings;
4
5 "AAC\nGTT" =~ /^.*$/;
6 print "Without /m:\n", $&, "\n";
7 #Without /m:
8 #Use of uninitialized value $& in print at
   XXX.pl line N.
9
10 "AAC\nGTT" =~ /^.*$/m;
11 print "With /m:\n", $&, "\n";
12 #With /m:
13 #AAC
```



# 模式匹配 | 修饰符 | /s

```
1 #!/usr/bin/perl
2
3 use warnings;
4
5 "AAC\nGTT" =~ /^.*$/;
6 print "Without /s:\n", $&, "\n";
7 #Without /s:
8 #Use of uninitialized value $& in print at
   XXX.pl line N.
9
10 "AAC\nGTT" =~ /^.*$/s;
11 print "With /s:\n", $&, "\n";
12 #With /s:
13 #AAC
14 #GTT
```





# 模式匹配 | 捕获

```
1 #!/usr/bin/perl
2
3 use strict; use warnings;
4
5 my $alphabet = join "", 'a' .. 'z';
6 $alphabet =~ /k(lmnop)q/;
7 print $1, "\n\n";
8 #lmnop
9
10 $alphabet =~ /(((a)b)c)/;
11 print "First pattern = ", $1, "\n";
12 print "Second pattern = ", $2, "\n";
13 print "Third pattern = ", $3, "\n";
14 #First pattern = abc
15 #Second pattern = ab
16 #Third pattern = a
```



# 模式匹配 | 捕获

```
1 my $string = "File code=123 name=test.txt";
2 if ($string =~ /code=(\d+)\s+name=( [\w\.] +) /) {
3     print "Code is $1\nName is '$2'\n";
4 }
```

```
1 my %q_count;
2 while (<>) {
3     while (/ \b (Q \w +) \b /g) {
4         ++$q_count{$1};
5     }
6 }
7 foreach my $word (sort {$q_count{$b} <=> $q_count{$a}
8     } keys %q_count) {
9     print "The word $word appeared ", $q_count{
10         $word}, " times\n";
11 }
```



## 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`



```
1 my $string = "xxxxxxhixxxxxxxxxxhixxxxxxxxhi";
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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl



# 读取文件 | tell & seek

```
1 for (;;) {  
2     for ($curpos = tell(FILE); $_ = <FILE>;  
3         $curpos = tell(FILE)) {  
4         # search for some stuff and put it  
5         into files  
6     }  
7     sleep($for_a_while);  
8     seek(FILE, $curpos, 0);  
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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl



# 文件夹处理 | 递归

```
1 #!/usr/bin/perl
2 use strict; use warnings;
3 list_recursively('pdb');
4
5 sub list_recursively {
6     my ($directory) = @_ ;
7     my @files = ();
8     unless ( opendir( DIRECTORY, $directory ) ) {
9         print "Cannot open directory $directory!\n";
10        exit;
11    }
12    @files = grep ( !/^\.\.?$/ , readdir(DIRECTORY) );
13    closedir(DIRECTORY);
14    foreach my $file ( @files ) {
15        if ( -f "$directory/$file" ) {
16            print "$directory/$file\n";
17        }
18        elsif ( -d "$directory/$file" ) {
19            list_recursively("$directory/$file");
20        }
21    }
22 }
```



```
1 #!/usr/bin/perl
2
3 use strict;
4 use warnings;
5 use File::Find;
6 #perldoc File::Find
7
8 find( \&my_sub, ('pdb') );
9
10 sub my_sub {
11     -f and ( print $File::Find::name, "\n" );
12 }
```



# 文件夹处理 | 通配

```
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 }
```



# 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 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})
6             (.{8}).{22}(..)/);
7
8     $n      =~ s/^\\s*//;
9     $element =~ s/^\\s*//;
10
11     if (($n == 1) or ($n == 1078)) {
12         printf "%8.3f%8.3f%8.3f %2s\\n", $x, $y,
13             $z, $element;
14     }
15 }
```



# 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!
```



```
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 = '
   AAAAAACCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGGGGGTTTTTTTTTTTTTTTTTTTT
   ';
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;
```



```
1 >A0000      Highly weird DNA.  This DNA is so un...  
2 AAAAAACCCCCCCCCCCCCCGGGGGGGGGGGGGGGGGGGGGGGGGTTTTTTTT  
3 TTTTTTTTTTTTTT
```



# 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl



# 运行外部程序

```
1 my $filename = $ARGV[0];
2 my $stride = '/usr/local/bin/stride';
3 my $options = '';
4 # 捕获输出
5 my @results = `$stride $options $filename`;
6 my $now = `date`;
7 my @functions = qw{ int rand length };
8 my %about;
9 foreach (@functions) {
10     # $about{$_} = `perldoc -t -f $_`;
11     $about{$_} = qx(perldoc -t -f $_);
12 }
13
14 # 不捕获输出, 返回值为程序退出状态
15 system "$stride $options $filename";
16 system 'date';
17 system 'tar', 'cvf', $starfile, @dirs;
```



# 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 BioPerl



# 浮点数比较

```
1 #!/usr/bin/perl
2
3 if ( 10 / 3 == ( ( 1 / 3 ) * 10 ) ) {
4     print "Success!\n";
5 }
6 else { print "Failure!\n"; }
7 #Failure!
8
9 if ( abs( 10/3 - ( ( 1/3 ) * 10 ) ) < 1e-10 ) {
10     print "Right!\n";
11     print "E=", abs(10/3 - ( (1/3) * 10 ) ), "\n";
12 }
13 else { print "Wrong!\n"; }
14 #Right!
15 #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 运行外部程序
- 7 浮点数比较
- 8 引用**
- 9 BioPerl



```
1 my @array = (1, 2, 3, 4);
2 my $slow_arrayref = \@array;
3 my $quick_arrayref = [1, 2, 3, 4];
4
5 my %hash = (
6     dog => 'woof',
7     cat => 'meow',
8 );
9 my $slow_hashref = \%hash;
10 my $quick_hashref = {
11     dog => 'woof',
12     cat => 'meow',
13 };
```





```
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',
9     currency => 'Euros',
10 );
11 my %ireland_info = ( ... );
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";
```



```
1 my %countries = (  
2     uk => {  
3         population => 60441457,  
4         language => 'English',  
5         currency => 'Pounds',  
6     },  
7     france => {  
8         population => 60656178,  
9         language => 'French',  
10        currency => 'Euros',  
11    },  
12    ireland => {  
13        ...  
14    },  
15 );  
16  
17 print "Population of France is ",  
18     $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";
```



```
1 my @experiments = (  
2     {  
3         sample_id => 1,  
4         sample_name => 'kidney',  
5         sample_measures => [12,56,34,65,76],  
6     },  
7     {  
8         sample_id => 4,  
9         sample_name => 'liver',  
10        sample_measures => [24,66,12,17,26],  
11    },  
12    { ... },  
13 );  
14  
15 foreach my $expt (@experiments) {  
16     print "The first measure for sample ",  
17         $expt->{sample_id},  
18         " (", $expt->{sample_name}, ") was ",  
19         $expt->{sample_measures}->[0], "\n";  
20 }
```



# 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 运行外部程序
- 7 浮点数比较
- 8 引用
- 9 **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 }
```



```
1 use strict; use Bio::SeqIO;
2 my $input_file = shift;
3 my $seq_in = Bio::SeqIO->new( -format => 'embl',
4                               -file   => $input_file,
5                               );
6 my @seq_array;
7 while ( my $seq = $seq_in->next_seq() ) {
8     push(@seq_array,$seq);
9 }
10 @seq_array = sort { $a->length <=> $b->length }
    @seq_array;
11 my $total = 0;
12 my $count = 0;
13 for my $seq ( @seq_array ) {
14     $total += $seq->length;
15     $count++;
16 }
17 print "Mean length ", $total/$count, " Median ",
18       $seq_array[$count/2]->length, "\n";
```



```
1 use Bio::SeqIO;
2
3 my $usage = "x2y.pl infile informat outfile outformat ";
4 my $infile = shift or die $usage;
5 my $informat = shift or die $usage;
6 my $outfile = shift or die $usage;
7 my $outformat = shift or die $usage;
8
9 my $seq_in = Bio::SeqIO->new( -file    => "$infile",
10                               -format => $informat,
11                               );
12 my $seq_out = Bio::SeqIO->new( -file    => ">$outfile",
13                               -format => $outformat,
14                               );
15 while (my $inseq = $seq_in->next_seq) {
16     $seq_out->write_seq($inseq);
17 }
```



```
1 use Bio::SeqIO;
2 my $usage = "splitgb.pl infile "; my $infile = shift or die $usage;
3 my $inseq = Bio::SeqIO->new( -file => "<$infile", -format => 'Genbank',);
4 my %outfiles = ( human => {
5     Genbank => Bio::SeqIO->new(
6         -file => '>human.gb',
7         -format => 'Genbank',
8     ),
9     Fasta   => Bio::SeqIO->new(
10        -file => '>human.fa',
11        -format => 'Fasta',
12    ),
13 },
14 other => {
15     Genbank => Bio::SeqIO->new(
16         -file => '>other.gb',
17         -format => 'Genbank',),
18     Fasta   => Bio::SeqIO->new(
19         -file => '>other.fa',
20         -format => 'Fasta',),
21 },
22 );
23 while (my $seqin = $inseq->next_seq) {
24     if ($seqin->species->binomial =~ m/Homo sapiens/) {
25         $outfiles{'human'}->{'Genbank'}->write_seq($seqin);
26         $outfiles{'human'}->{'Fasta'}->write_seq($seqin);
27     } else {
28         $outfiles{'other'}->{'Genbank'}->write_seq($seqin);
29         $outfiles{'other'}->{'Fasta'}->write_seq($seqin);
30     }
31 }
```



```
1 use Bio::DB::GenBank;
2 use Bio::DB::Query::GenBank;
3
4 $query = "Arabidopsis[ORGN] AND topoisomerase[TITL] and
   0:3000[SLEN]";
5 $query_obj = Bio::DB::Query::GenBank->new(-db => '
   nucleotide',
6                                           -query =>
   $query );
7
8 $gb_obj = Bio::DB::GenBank->new;
9
10 $stream_obj = $gb_obj->get_Stream_by_query($query_obj);
11
12 while ($seq_obj = $stream_obj->next_seq) {
13     # do something with the sequence object
14     print $seq_obj->display_id, "\t", $seq_obj->length,
       "\n";
15 }
```



```
1 #!/usr/bin/perl
2 use strict; use warnings;
3 use Bio::DB::Query::GenBank; use Bio::DB::GenBank;
4 use Bio::SeqIO;
5
6 my $query_string = $ARGV[0];
7 my $fo_fa=$query_string.".fa"; my $fo_gb=$query_string.".gb";
8 my $query = Bio::DB::Query::GenBank->new(
9     -db => 'nucleotide', -query => $query_string
10 );
11 my $gb = Bio::DB::GenBank->new;
12 my $stream = $gb->get_Stream_by_query($query);
13
14 my %outfiles = (
15     Fasta => Bio::SeqIO->new( -file => ">$fo_fa", -format => '
16     Fasta',),
17     Genbank => Bio::SeqIO->new( -file => ">$fo_gb", -format => '
18     Genbank',),
19 );
20 while ( my $seq = $stream->next_seq ) {
21     $outfiles{'Fasta'}->write_seq($seq);
22     $outfiles{'Genbank'}->write_seq($seq);
23 }
```





TEX

LATEX

X<sub>Y</sub>TEX

Beamer

