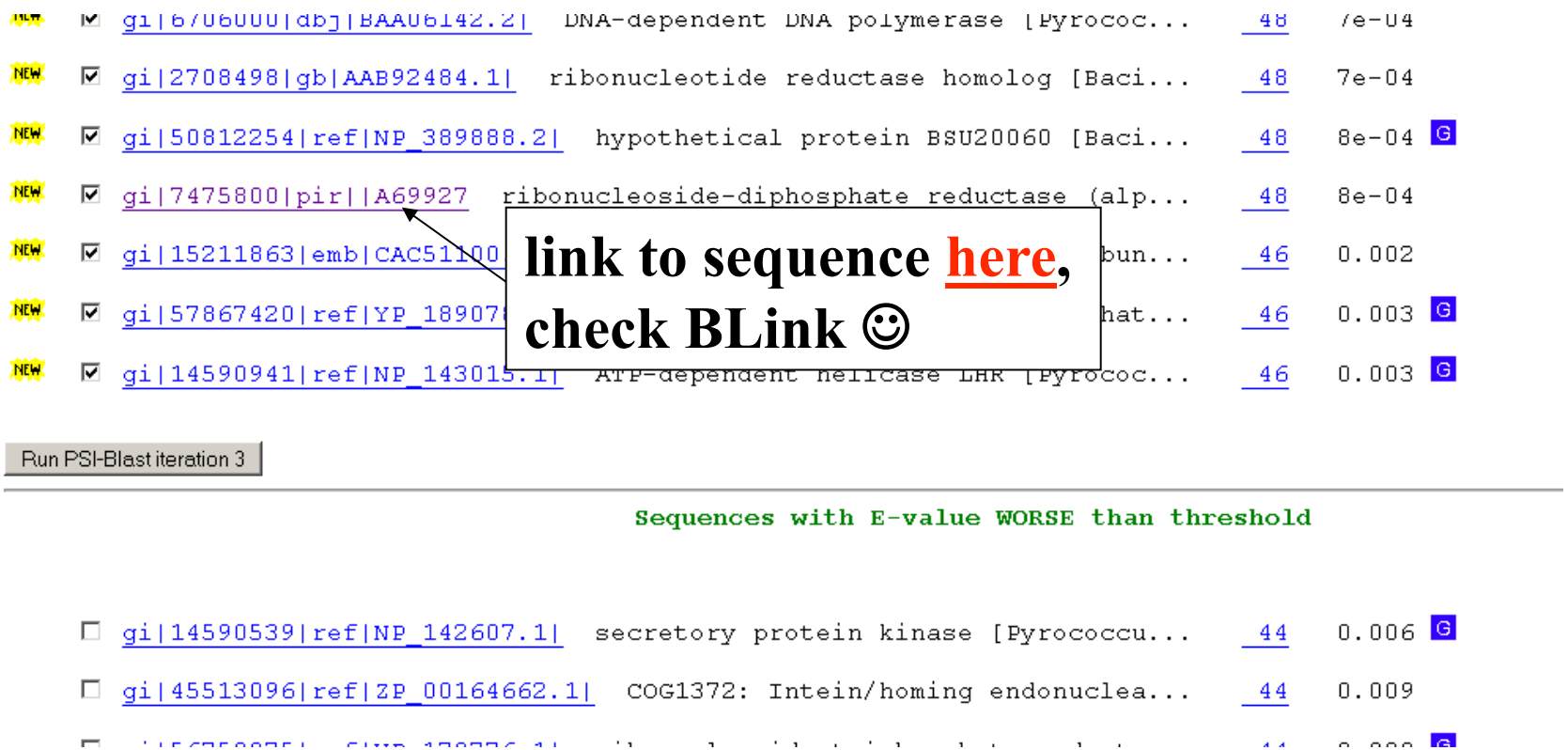
MCB 5472

Psi BLAST,
Perl: Arrays, Loops

J. Peter Gogarten
Office: *BPB 404*
phone: *860 486-4061,*
Email: *gogarten@uconn.edu*

Psi-Blast Results Query: 55670331 (intein)



PSI BLAST and E-values!

Psi-Blast is for finding matches among divergent sequences (position-specific information)

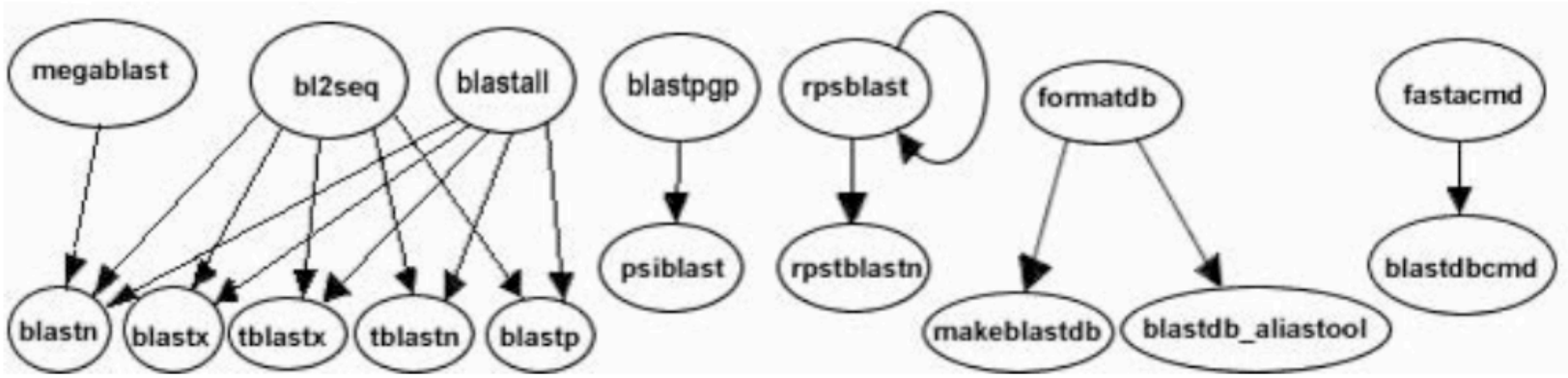
WARNING: For the nth iteration of a PSI BLAST search, the E-value gives the number of matches to the profile NOT to the initial query sequence! The danger is that the profile was corrupted in an earlier iteration.

The NCBI has released a new version of blast. The command line version is blast+ . The new version is faster and allows for more flexibility, but at present we still have problems with running it on the cluster.

The new commands are equivalent to the blastall commmands:

Functionality offered by BLAST+ applications

The functionality offered by the BLAST+ applications has been organized by program type, as to more closely resemble Web BLAST. The following graph depicts a correspondence between the NCBI C Toolkit BLAST command line applications and the BLAST+ applications:



The legacy_blast.pl script that is part of blast+ translates blastall commands into the blast+ syntax. E.g.:

```
$ ./legacy_blast.pl megablast -i query.fsa -d nt -o mb.out --print_only
/opt/ncbi/blast/bin/blastn -query query.fsa -db "nt" -out mb.out
$
```

From the blast+ manual:

> The easiest way to get started using these command line applications is by means of the legacy_blast.pl PERL script which is bundled along with the BLAST+ applications. To utilize this script, simply prefix it to the invocation of the C toolkit BLAST command line application and append the --path option pointing to the installation directory of the BLAST+ applications. For example, instead of using
>
> ```
> blastall -i query -d nr -o blast.out
> ```
>
> use
>
> ```
> legacy_blast.pl blastall -i query -d nr -o blast.out
> --path /opt/blast/bin
> ```

3) Write a short Perl script that calculates the circumference of a circle given a radius provided by the user.

```
#!/usr/bin/perl -w
use strict;
print "This program finds the circumference of a circle.\n";
print "What is your radius?\n";
chomp (my $radius = <STDIN>);
print "The circumference of a circle with radius of $radius is\n";
print 2*3.141592654*$radius."\n"; #Equation for circle circumference
```

```
#!/usr/bin/perl -w
#As usual there are 1000 ways to do this.
#one is to define $pi or the constant PI, eg. as follows
#use constant PI => 4*atan2(1,1);
#or use a module
use Math::Trig; #allows to use the Math::Trig module that is part of perl
$circumference=0; #reset variables
print "\nEnter radius:";
chomp (my $radius=<>);
$circumference= $radius*pi*2;
print "\nwith radius=$radius ,\nthe circumference is $circumference\n\n";
```

The best way to find which module to use is google. You can search core modules at http://perldoc.perl.org/search.html?

Old Assignment for Monday

1) Write a 2 sentence outline for your student project
2) Read chapter P5 and P12 conditional statements and on "for, foreach, and while" loops.
   http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v2.3.3.pdf

Background:

```
@a=(0..50);
#  This assigns numbers from 0 to 50 to an array,
#   so that $a[0] =0; $a[1] =1; $a[50] =50
```

3) Write perl scripts that add all numbers from 1 to 50. Try to do this using at least two different control structures.
4) Create a program that reads in a sequence stored in a file handed to the program on the command line and determines GC content of a sequence. Use class3.pl as a starting point.

Go through class3.pl script (http://gogarten.uconn.edu/mcb5472_2010/class3.pl).

If time go through Olga's search for distant homologs webpage at (use cd00081for PSSM viewer)
http://www.mta.ca/~ozhaxybayeva/bioinf2010/class10.html

%GC counter, part A: read in seqs

```
#!/usr/bin/perl -w
use strict;
##########INPUT Sequence, concatenated into a single string##########
#skip annotation lines in case of fasta. if multiple annotation lines, concatenate these too.
#
unless(@ARGV==1) {die "please provide name of the file in the command line!!\n";}
my$filename=$ARGV[0]; #takes filenname from input line
open(IN, "< $filename") or die "cannot open $filename:$!"; #assigns filehandle IN to filename or dies

my$seq=''; #assigns empty string
my$line='';
my$name='';
my@bases=(); #assigns empty list
while(defined($line=<IN>)){

        chomp($line);
        if ($line=~/^>/) {  #look for beginning of line starting with > (^ is an anchor for the beginning of
                $name .= $line;
                }
        else {
                $seq .= $line ;
                }
}
```

%GC counter, part B: move seqs to array

```
#################### move sequence to array
#  check for all CAPS, report non ATGCs, remove white spaces
#
$seq =~ tr/atgc/ATGC/; #translates all ATGC to upper case
$seq =~ s/\s//g;# substitutes all white spaces \s with nothing globally in $seq
@bases=split(//,$seq);  #splits string into separate elements (bases)

my$num_bases=@bases; #length of array
```

%GC counter, part B: calculate %GC

```
###################calculate GC content
my$num_GC=0;

for (my $i=0; $i<($num_bases); $i++)  #counts Gs and Cs in @bases Note the number of bases is one larger than the arr
        {
        if(($bases[$i]=~"G") or ($bases[$i]=~"C")) #if it matches G or C increase counter
                {$num_GC++;
                }
        if (!(($bases[$i]=~"G") or ($bases[$i]=~"A") or ($bases[$i]=~"T") or ($bases[$i]=~"C")))
                {print "Warning there is a strange base $bases[$i] before position $i\n";
                my$errors++;}
        }

if (defined (my$errors)){$num_bases=$num_bases-$errors};
my $GC_content=($num_GC/$num_bases)*100;
print "\nThe GC content of the sequence in the file ".'"'."$filename".'"'. " is $GC_content\%.\n\n";
if (!($name eq '')) {print "Annotation line(s) in $filename was/were $name\n";}
```

Control structures: Sum 1..50

while ( ) { }

```
#!/usr/bin/perl -w
$sum=0;
$count=0;

while ($count <50) {
    $count++; #this is tricky in the last loop $count is 49 and then increased to 50 and added
    $sum += $count;
    };
print "$sum\n"
```

for (, ,) { }

```
#!/usr/bin/perl/
$sum=0;
$count=0;

for ($count =0; $count < 51; $count++)  {
    #$sum=$sum+$count;
    $sum += $count
};
print "$sum\n"
```

Control structures: Sum 1..50

```
#!/usr/bin/perl/
$sum=0;
@array = (1..50);
foreach (@array)  {
     #$sum=$sum+$_;
     $sum += $_;
};
print "$sum\n"
```

foreach ( ) { };

```
#!/usr/bin/perl/
$sum=0;
$count=0;
while () {
     $sum += $count;
     $count+=1;
     if ($count >50) {last};
}
print "$sum\n"
```

Infinite loop with last:

while () {

if( ) {last};
};

Control structures: Sum 1..50

```
#!/usr/bin/perl
$sum=0;
@array = (1..50);
$count=0;
while (defined($array[$count]))
{
    $sum += $array[$count];
    $count += 1;
    #print "$array[$count]\t $sum\n";
};
print "$sum\n"
```

while (defined ( )) { };

```
#!/usr/bin/perl -w
$sum=0;
@array = (0..50);
$count=0;
for ($count=1; ($count<51); $count++){
    $sum += $array[$count];
    #$temp=$array[$count];
    #print "\$count=$count sum is $temp\t $
    }
print "$sum\n";
```

for (, ,) { }

Counting elements of an array

Could have started at 0

For Monday

Write a script that reads in a sequence and prints out the reverse complement.
Modify your script to that it can handle a sequence that goes over several lines.

•Background: `$comp =~ tr/ATGC/TACG/;`
#translates every A in $comp into a T; every T into an A; every G into a C and every C into a G

•Read P 14 on hashes, write the program suggested in the chapter.

For Monday

Do the following statements evaluate to true or false? (Check P5)

- 1
- 0 && 1
- 0||1
- 45
- 45-45
- 45/45
- 45==45
- 45<=>45
- 45<=50
- 55>=50
- 50<=>70
- 45!=45
- 45!=50

Operator	Meaning	Example
==	equal to	if ($x == $y)
!=	not equal to	if ($x != $y)
>	greater than	if ($x > $y)
<	less than	if ($x < $y)
>=	greater than or equal to	if ($x >= $y)
<=	less than or equal to	if ($x <= $y)
<=>	comparison	if ($x <=> $y)

from http://korflab.ucdavis.edu/Unix and Perl/unix and perl v2.3.3.pdf

String comparison operators in Perl

Operator	Meaning	Example
eq	equal to	if ($x eq $y)
ne	not equal to	if ($x ne $y)
gt	greater than	if ($x gt $y)
lt	less than	if ($x lt $y)
.	concatenation	$z = $x . $y
cmp	comparison	if ($x cmp $y)

from http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v2.3.3.pdf

Most of the smaller assignments should be solvable within half an hour. Using the notes, the text book and the internet try to solve one problem for not more than one hour. Then ask me or Tim for help.

In total, the assignments for one week might take a few hours, but if it goes beyond 6 hours total, ask for help, or hand in the latest version of your attempt to solve the assignment. Sometimes, a little help can go a long way. The main reason for the assignments is to make you actually write code and to learn form the mistakes you make.

Hashes are tables that relate keys and values.
(in the array the number of the field could be considered the key:
@a=(1..51) => $a[0]=1, $a[50]=51)

In a %ash the entry for the key is the address where the value is stored.
E.g., you could have a hash where the students age is stores as value and the student ID is the key.
But you also could use the students name as key and the ID or age or …. as value. This works very economically, especially if the table is sparse.

```
my (%studentID, %student_first_name, %studentGPA);
$studentID{gogarten}=9999;
$student_first_name{gogarten}='Johann Peter';
$studentGPA{gogarten}=3.2;
```

In many instances you need to make sure that the key you want to uses has not yet been assigned. `If (exists ($studentID{gogarten}) {};`

Go through class 4.pl

http://gogarten.uconn.edu/mcb5472_2010/class4.pl
http://gogarten.uconn.edu/mcb5472_2010/gi_list.txt