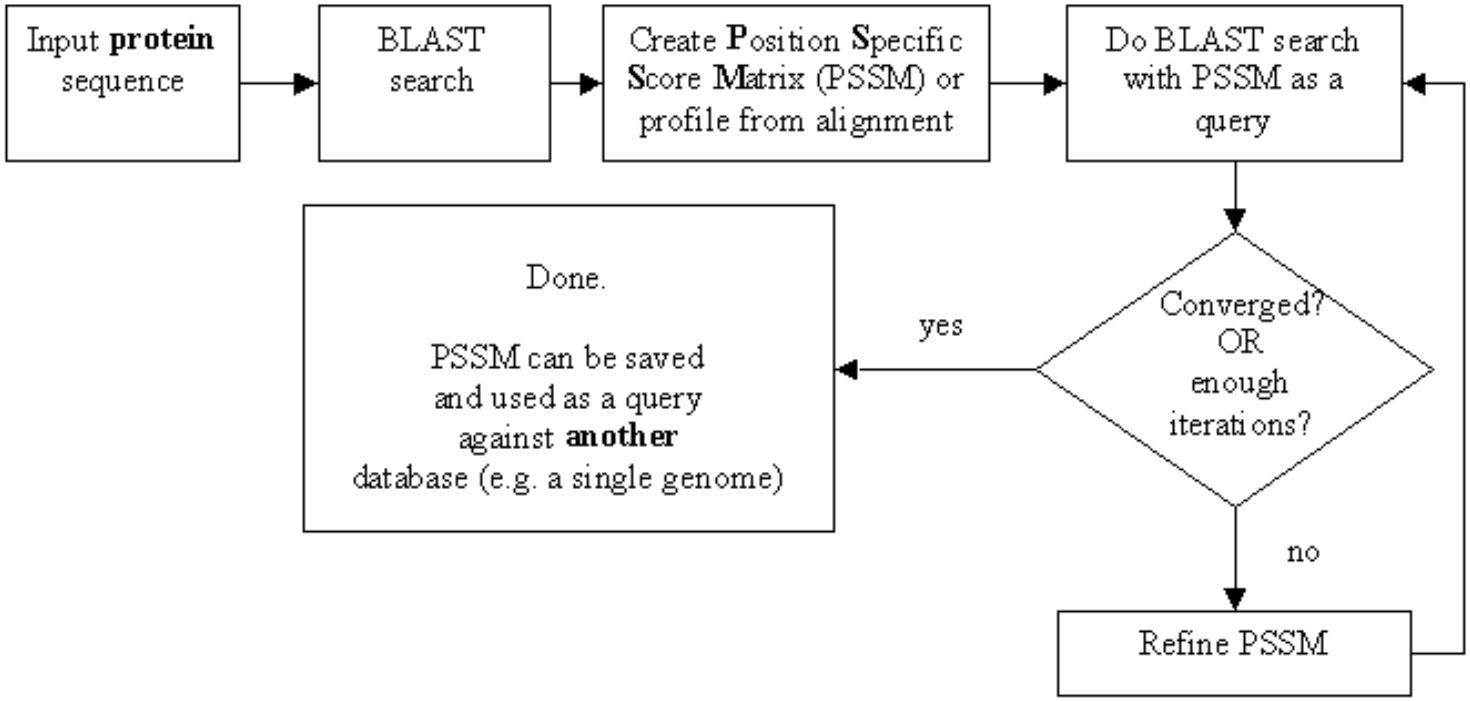
MCB 5472

Psi BLAST,
Perl: Arrays, Loops

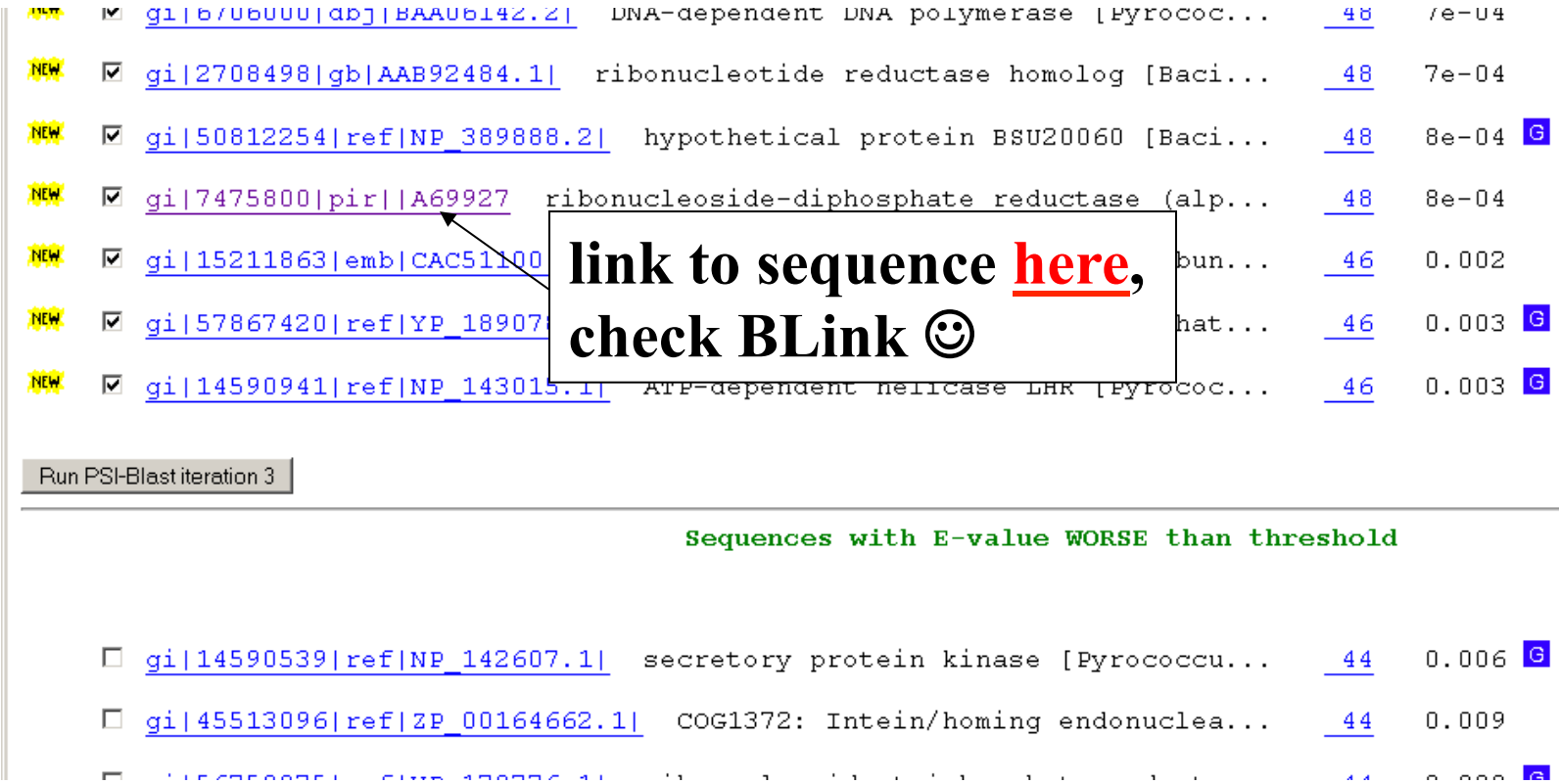
J. Peter Gogarten
Office: *BPB 404*
phone: *860 486-4061,*
Email: *gogarten@uconn.edu*

- Blast: find High Scoring Pairs between query and database of target sequences
- Join neighboring HSPs into single gapped alignment (formerly known as gapped blast)
- Report all HSPs (joined or not) between query and target sequence
- Note: if your target is a single genome, all HSPs (joined or not) will be reported as a single match
- Position Specific Iterated Blast: Find matches between PSSM and target
- RPS (Reverse PSI) blast: find matches between query and database of PSSMs

PSI BLAST scheme



Psi-Blast Results Query: 55670331 (intein)



PSI BLAST and E-values!

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

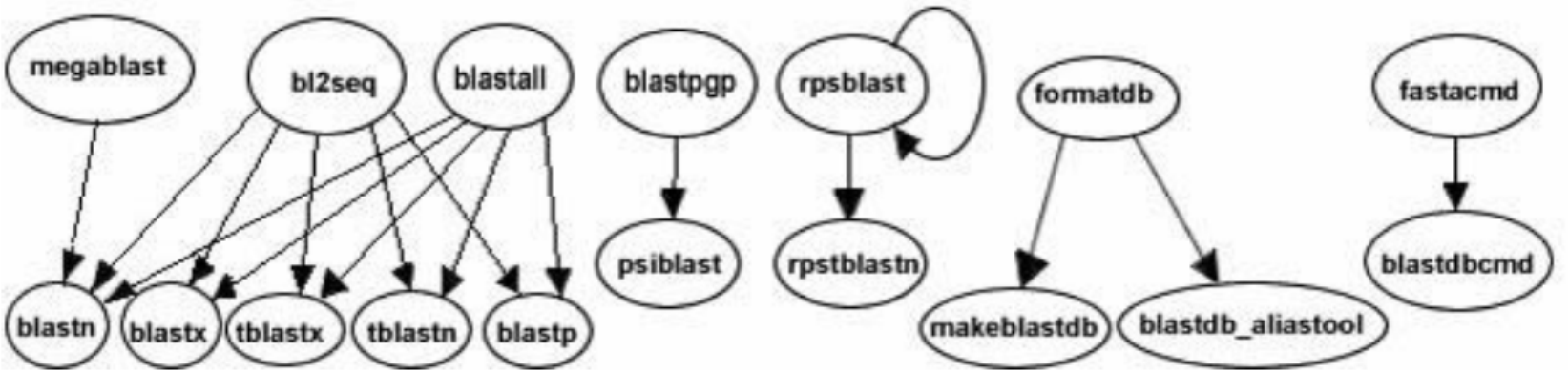
WARNING: For the nth iteration of a PSI BLAST search, the E-value gives the number of matches to the profile NOT to the initial query sequence! The danger is that the profile was corrupted in an earlier iteration.

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
> ```

PSI Blast from the command line

Often you want to run a PSIBLAST search with two different databanks -
one to create the PSSM, the other to get sequences:
To create the PSSM:

```
blastpgp -d nr -i subI -j 5 -C subI.ckp -a 2 -o subI.out -h 0.00001 -F f
```

```
blastpgp -d swissprot -i gamma -j 5 -C gamma.ckp -a 2 -o gamma.out -h 0.00001 -F f
```

Runs a **4 iterations** of a PSIblast
the -h option tells the program to use matches with E <10^-5 for the next iteration,
(the default is 10^{-3})
-C creates a checkpoint (called subI.ckp),
-o writes the output to subI.out,
-i option specifies input as using subI as input (a fasta formated aa sequence).
The nr databank used is stored `in /common/data/`
-a 2 use two processors

(It might help to use the node with more memory (017)
(command is `ssh node017`)

To use the PSSM:

```
blastpgp -d /Users/jpgogarten/genomes/msb8.faa -i subI -a 2 -R
subI.ckp -o subI.out3 -F f

blastpgp -d /Users/jpgogarten/genomes/msb8.faa -i gamma -a 2 -R
gamma.ckp -o gamma.out3 -F f
```

Runs another iteration of the same blast search, but uses the databank `/Users/jpgogarten/genomes/msb8.faa`

-R tells the program where to resume
-d specifies a different databank
-i input file - same sequence as before
-o output_filename
-a 2 use two processors

PSI Blast and finding gene families within genomes

use PSSM to search genome at the nucleotide level:

A) Use protein sequences encoded in genome as target:

```
blastpgp -d target_genome.faa -i query.name -a 2 -R query.ckp -o
    query.out3 -F f
```

B) Use nucleotide sequence and tblastn. This is an advantage if you are also interested in pseudogenes, and/or if you don't trust the genome annotation:

```
blastall -i query.name -d target_genome_nucl.ffn -p psitblastn -R
    query.ckp
```

Assignment 1: blastall

Histogram script replaced: working verison in scripts and linked in assignments #2

Do example on data in blasttest

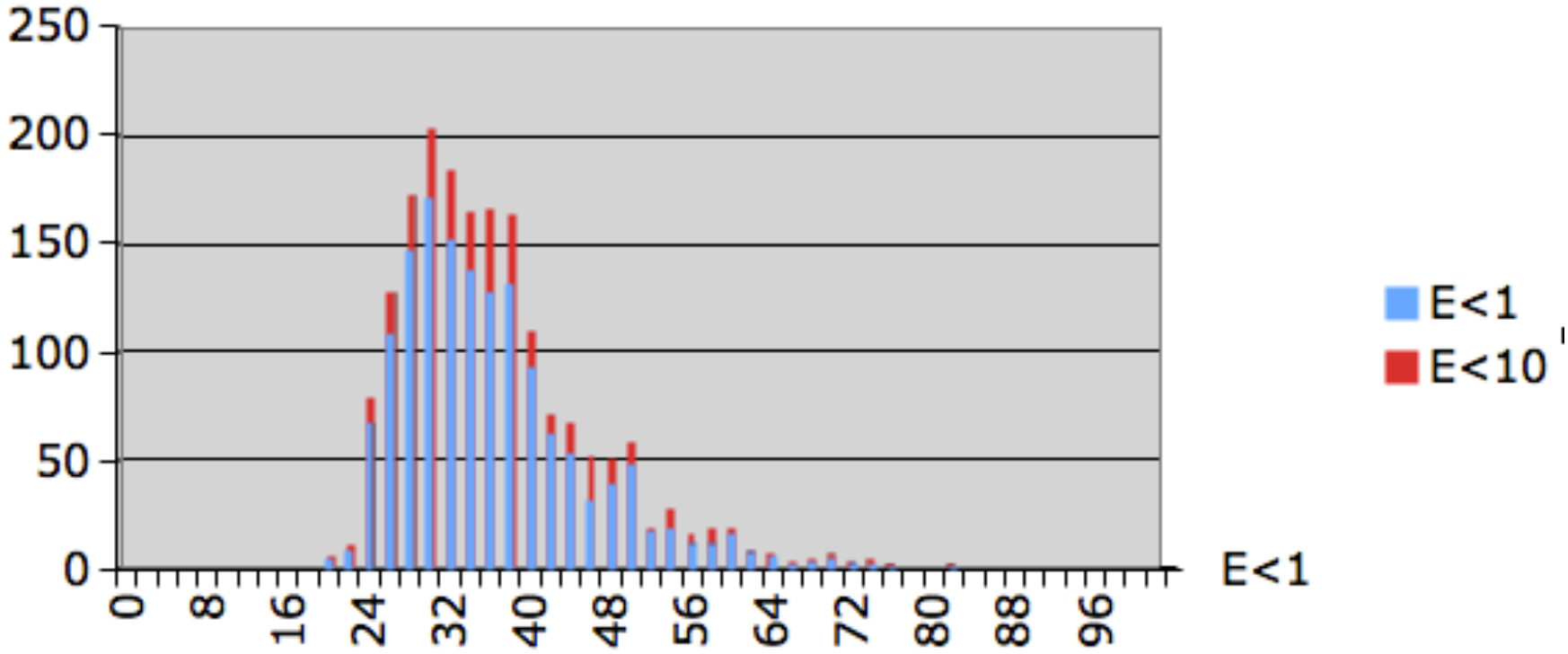
Go through output –m9 and –m8

Go through extract_lines.pl

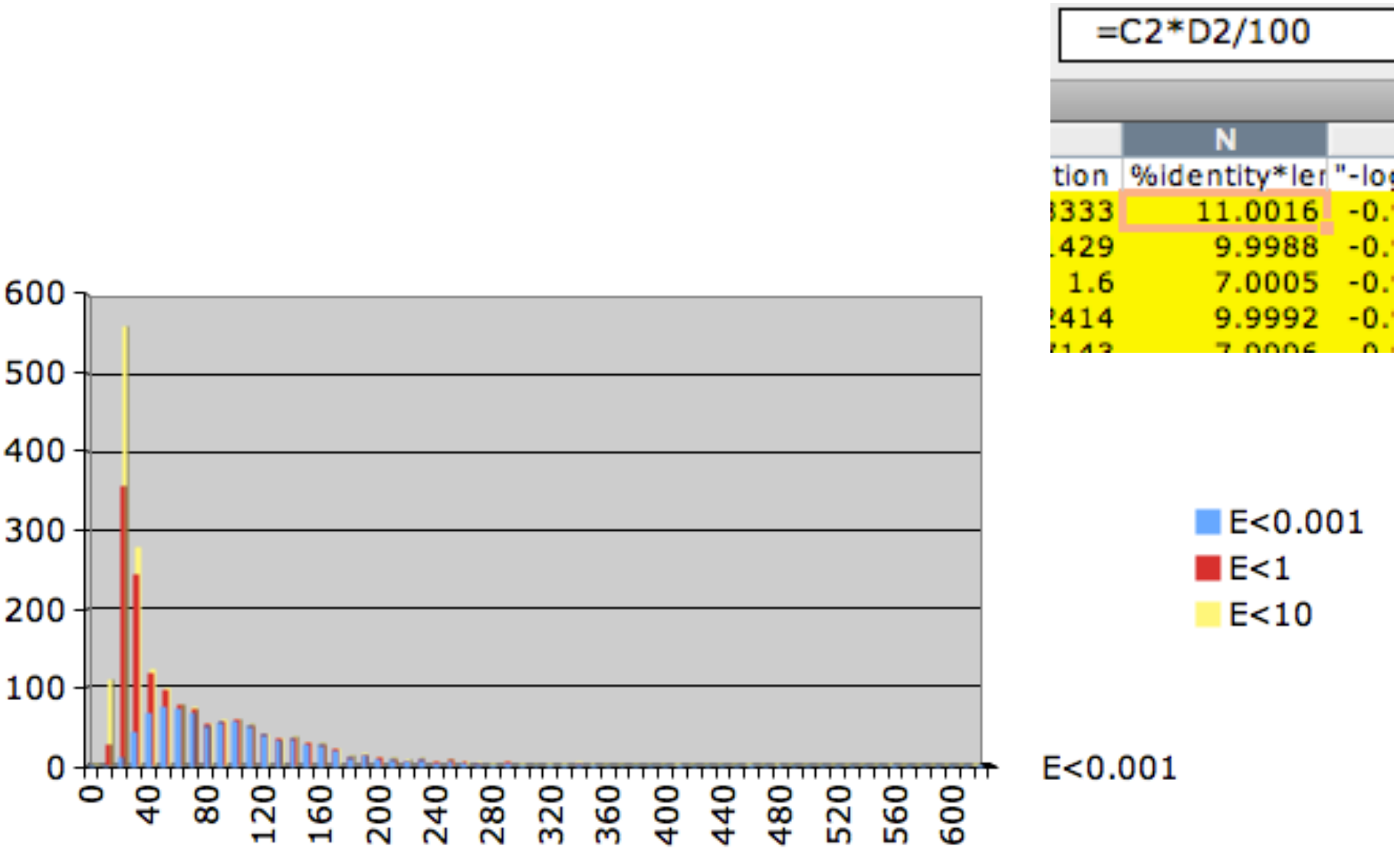
Open in Excel, save columns -> make histograms in R

Open workbook2 in 2004 Excel -> check %identity and # of identical residues

Histogram of percent identity for significant and insignificant hits



Number of identical residues for significant and insignificant hits



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

5)

For the following array declaration

@myArray = ('A', 'B', 'C', 'D', 'E');

what is the value of the following expressions:

```
$#myArray
length(@myArray)
$myArray[1]
$n=@myArray
reverse (@myArray)
```

Go through array-scalar example

```
#! /usr/bin/perl
#use strict;
#use warnings;
my @test=(); #() is the empty list, resets the array
my $counter = ''; # empty string, may be preferrable over zero;

print "\nbefore assignment:\n";
print "values in \@test are";
print @test; #should print nothing
print "\n";
print "\n value of \$counter= $counter \n";

@test=(1..50); #fills the array
print "\n after assignment:\n";
print "\nfirst element of array \@test is $test[0], it is stored in slot zero\n";

$counter=$#test;
print "the last element if the array is stored in slot number $counter \n";
$counter=@test; #using the array in scalar context returns the size of the array
print "\nthe array \@test has $counter elements after the first assignment\n";

$test[100]=99;
$counter=@test; #using the array in scalar context returns the size of the array
print "\nthe array \@test has $counter elements after the second assignment\n";
print join (" ",@test);
print"\n";
~
~
"array_scalar example.pl" 26L, 934C
```

```
node008:~/perl2012/class04 jpgogarten$ perl array_scalar\ example.pl

before assignment:
values in @test are

 value of $counter=

 after assignment:

first element of array @test is 1, it is stored in slot zero
the last element if the array is stored in slot number 49

the array @test has 50 elements after the first assignment

the array @test has 101 elements after the second assignment
1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
                                                                  99
```

```perl
#!/usr/bin/perl -w
use strict;
my @myArray = ('A', 'B', 'C', 'D', 'E');
print $#myArray;
print "\n";
print length(@myArray);
print "\n";
print $myArray[1];
print "\n";
print my $n=@myArray;
#using the array in a scalar context returns the number of fields
print "\n";
print reverse (@myArray);
print "\n";
```

```
node008:~/perl2012/class04 jpgogarten$ perl myArrayEx.pl
4
1
B
5
EDCBA
```

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.

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

6)
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.
Details in class3.pl. See the challenge!

%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 beginni
                $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); $i++)  #counts Gs and Cs in @bases Note the number of bases is one larger than t
        {
        if(($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";}
```

Challenge: GC counter in rolling window

```
my$num_bases=@bases;   #length of array
my@window=();
my@GC_content=();
my$count=0;
my$num_CG=0;
################### assign first window
for (my $k = 0; $k < 100; $k++)
        {
        $window[$k] = $bases[$k];
        };

###################calculate GC content
for (my$l=100; $l<($num_bases+1); $l++) { #big loop starts


        for (my $i=0; $i<(100); $i++)
        # counts Gs and Cs in window Note the number of bases is one larger than the array
                {
                if(($window[$i]=~"G") or ($window[$i]=~"C")) #if it matches G or C increase
                        {$num_GC++;}
                }
        $GC_content[$count]=($num_GC);
        # print "$num_GC $bases[$l] ";
        $num_GC=0;
#move window by one to right
        $count++;
        shift @window;
        # print "$test\n";
        push @window, $bases[$l];
        }
print join("   ",@GC_content);
print "\n";
```

For Next 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 Kristen, or Erica for help.

In total, the assignments for one week might take a few hours, but if it goes beyond 5 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 stored 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}="nd";
```

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

Go through class 4.pl

http://gogarten.uconn.edu/mcb5472_2012/class4.pl
http://gogarten.uconn.edu/mcb5472_2012/gi_list.txt