LAB MANUAL
BIOINFORMATICS LABORATORY (BT2308)
V SEMESTER

B.TECH DEGREE PROGRAMME
Department of Biotechnology

RAJALAKSHMI ENGINEERING COLLEGE
THANDALAM
CHENNAI

Prepared by

Mr. N.Chakravarthy
&

Mrs.P.Madhumitha

LIST OF EXPRIMENTS

1. Introduction to UNIX basic commands and UNIX Filters.
2. Perl programming and applications to Bioinformatics.
- Basic scripting.
- Regular expressions.
- File i/o & control statement.
- Subroutines & functions.
- Writing scripts for automation.

3. Types of Biological Databases and Using it.
- Genbank.
- Protein Data Bank .
- Uniprot.

4. Sequence Analysis Tools
- Use of BLAST, FASTA (Nucleic Acids & Protiens).
- Use of Clustal W.
- Use of EMBOSS.

5. Phylogenetic Analysis
- Use of Phyllip.

6. Molecular Modeling
- Homology Modeling – Swissmodeller.
- Any Open Source Software.

INTRODUCTION TO UNIX BASIC COMMANDS AND UNIX FILTERS.

Exp 1: **Basic UNIX commands**

Aim: To verify the basic unix Commands and Filters

1.DATE command
Display the server date and time
Syntax :$date
Output: Mon June 12, 2010

2.CALENDER command
Display the particular month calendar or year calendar
Syntax :$cal <*month name*> or <*year*>
Eg:$ cal 2010 – Prints the calendar for the entire year 2010.
$cal 11 2009-Display Nov-2009 calendar

3.ECHO command
It is used to print the message on the screen,whatever you happen to type on the line.
Syntax :$echo<*text to be displayed on the screen*>
Eg:$echo Welcome to Bioinformatics Laboaratory 2010

4.BANNER command
It is used to print the message in large letters to gibe the impression of a banner.
Syntax :$banner<*text*>
Eg:$banner BIOTECH

5.WHO command
Display the information about all users who have logged into the system currently.
Syntax :$who

6.WHO AM I command
It gives login details of a particular system i.e it gives the user name,terminal name,date and time of login
Syntax :$who am i

7.EXIT command
It is used to logout from the user sessions.
Syntax :$exit

8.CLEAR command
It is used to clear the screen.
Syntax :$clear

9.LOGNAME commnd
It is used to display the current user name.
Syntax :$logname

10.ID command
The id command is used to display the numerical value that corresponds to your login name i.e.,every valid UNIX user is assigned a login name,a user id and a group-id.

Syntax :$id

11.TTY command
The tty(teletype) command is used to know the terminal name that we are using.
Syntax :$tty

12 UNAME command
Display the name of the operating system.
Syntax :$uname

Exp 2: File and Directory Commands

Aim : To create the file(s) and verify the file handling commands.

1.TOUCH command
Create file(s) with zero byte size.
Syntax :$touch <*filename*>
Eg:$touch biotech
$touch bioinfo gene dna.

2.CAT command
Create file with data and display data in the file.
(a)File creation
Syntax :$cat ><*filename*>
Eg:$cat>test
……
……
……
Ctrl+d[to close file]
Eg:$cat>>test--->append data to the file 'test'
……
……
Ctrl+d
Eg:$cat file1 file2>file3 ---->data in file1,file2 copied to file3.
(b)Display data from the file(s)
Syntax :$cat <*filename*>
Eg:$cat file1---->display data from file1
Eg:$cat file1 file2 file3

3.CP command
It is used to copy the contents of one file to another file and copies the file from one place to another.
Syntax :$cp <*source filename*> <*destination filename*>
Eg:$cp file1 file2

4.MV command
It is used to rename the file(s).
Syntax :$mv <*old filename*> <*new filename*>
Eg:$mv file1 file4--->file1 renamed to file4

5.RM command

It is used to remove the file(s)

Syntax :$rm <*filename(s)*>

Eg:$rm file1--->file1 is removed

Eg:$rm file2 file3 file4--->remove three files

Eg:$rm*--->remove all files in current directory

6.FILE command

It is used to determine the type of the file.

Syntax :$file <*file name(s)*>

Eg:$file file1

Eg:$file*

7.WC command

This command is used to display the number of lines,number of words and number of characters in a file

Syntax :$wc <*file name(s)*>

Eg:$wc file1

Eg:$wc –l file1--->display only the number of lines in file1.

Eg:$wc –w file1--->display only the number of words in file1.

Eg:$wc –c file1--->display only the number of characters in file1.

Aim : To create the directorie(s) and verify the directory commands

1.PWD command

It is used to know the current working directory

Syntax :$pwd

2.MKDIR command

It is used to create an empty directory

Syntax :$mkdir <*directory name*>

Eg:$mkdir program

3.CD command

It is used to move from one directory to another directory

Syntax :$cd <*directory name*>

Eg:$cd biotech

4.RMDIR command

It is used to remove the directory only if it is empty

Syntax :$rmdir <*directory name*>

Eg:$rmdir program

Syntax :$rm –r <*directory name*>

Eg:$rm –r biotech

5.MV command

It is used to rename the directory

Syntax:$mv <*old directoryname*> <*new directoryname*>

Eg:$mv program biotech--->directory'program' renamed to 'biotech'

6.LS command

It is used to view the contents of the directory

Syntax :$ls

MODULE A: BASIC SCRIPTING

Exp 3 **Basic mathematical operations**

Aim:

To perform Basic mathematical operations using PERL

Program :

```
#!/usr/bin/perl -w
$x=10;
print"\tThe value of first variable,x is : $x\n";
$y=5;
print"\tThe value of second variable,y is : $y\n";
$sum=$x+$y;
print"\tThe sum of Two variables is : $sum\n";
$diff=$x-$y;
print"\tThe difference of Two variables is : $diff\n";
exit;
```

Output :

```
The value of first variable,x is : 10
The value of second variable,y is : 5
The sum of Two variables is : 15
The difference of Two variables is : 5
```

Exp 4 **Concatenating DNA**

Aim:

To Concatenating DNA sequences using PERL

Program :

```
#!/usr/bin/perl -w
# Concatenating DNA
# Store two DNA fragments into two variables called $DNA1 and $DNA2
$DNA1 = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
$DNA2 = 'ATAGTGCCGTGAGAGTGATGTAGTA';
# Print the DNA onto the screen
print "Here are the original two DNA fragments:\n\n";
print $DNA1, "\n";
print $DNA2, "\n\n";
# Concatenate the DNA fragments into a third variable and print them
# Using "string interpolation"
$DNA3 = "$DNA1$DNA2";
print "Here is the concatenation of the first two fragments (version 1):\n\n";
print "$DNA3\n\n";
# An alternative way using the "dot operator":
# Concatenate the DNA fragments into a third variable and print them
$DNA3 = $DNA1 . $DNA2;
print "Here is the concatenation of the first two fragments (version 2):\n\n";
print "$DNA3\n\n";
print "Here is the concatenation of the first two fragments (version 3):\n\n";
```

```
print $DNA1, $DNA2, "\n";
exit;
```

Output :

```
Here are the original two DNA fragments:
ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGTA
Here is the concatenation of the first two fragments
(version 1):
ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGTA
Here is the concatenation of the first two fragments
(version 2):
ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGTA
Here is the concatenation of the first two fragments
(version 3):
ACGGGAGGACGGGAAAATTACTACGGCATTAGCATAGTGCCGTGAGAGTGATGTAGTA
```

Exp 5 Transcribing DNA into RNA

Aim:
To Transcribe DNA sequence into RNA sequence using PERL

Program :

```
#!/usr/bin/perl -w
# Transcribing DNA into RNA
# The DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
# Print the DNA onto the screen
print "Here is the starting DNA:\n\n";
print "$DNA\n\n";
# Transcribe the DNA to RNA by substituting all T's with U's.
$RNA = $DNA;
$RNA =~ s/T/U/g;
# Print the RNA onto the screen
print "Here is the result of transcribing the DNA to
RNA:\n\n";
print "$RNA\n";
# Exit the program.
exit;
```

OutPut :

```
Here is the starting DNA:
ACGGGAGGACGGGAAAATTACTACGGCATTAGC
Here is the result of transcribing the DNA to RNA:
ACGGGAGGACGGGAAAAUUACUACGGCAUUAGC
```

Exp 6 Calculating the reverse complement of a strand of DNA

Aim:
To Calculate the reverse complement of a strand of DNA using PERL

Program :

```
#!/usr/bin/perl -w
# Calculating the reverse complement of a strand of DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
# Print the DNA onto the screen
print "Here is the starting DNA:\n\n";
print "$DNA\n\n";
# Calculate the reverse complement
# Warning: this attempt will fail!
# First, copy the DNA into new variable $revcom
# (short for REVerse COMplement)
# Notice that variable names can use lowercase letters like
# "revcom" as well as uppercase like "DNA". In fact,
# lowercase is more common.
#
# It doesn't matter if we first reverse the string and then
# do the complementation; or if we first do the
complementation
# and then reverse the string. Same result each time.
# So when we make the copy we'll do the reverse in the same
statement.
#
$revcom = reverse $DNA;
#
# Next substitute all bases by their complements,
# A->T, T->A, G->C, C->G
#
$revcom =~ s/A/T/g;
$revcom =~ s/T/A/g;
$revcom =~ s/G/C/g;
$revcom =~ s/C/G/g;
# Print the reverse complement DNA onto the screen
print "Here is the reverse complement DNA:\n\n";
print "$revcom\n";
#
# Oh-oh, that didn't work right!
# Our reverse complement should have all the bases in it,
since the
# original DNA had all the bases--but ours only has A and G!
#
# Do you see why?
#
# The problem is that the first two substitute commands
above change
# all the A's to T's (so there are no A's) and then all the
# T's to A's (so all the original A's and T's are all now
A's).
# Same thing happens to the G's and C's all turning into G's.
#
```

```
print "\nThat was a bad algorithm, and the reverse complement was wrong!\n";
print "Try again ... \n\n";
# Make a new copy of the DNA (see why we saved the original?)
$revcom = reverse $DNA;
# See the text for a discussion of tr///
$revcom =~ tr/ACGTacgt/TGCAtgca/;
# Print the reverse complement DNA onto the screen
print "Here is the reverse complement DNA:\n\n";
print "$revcom\n";
print "\nThis time it worked!\n\n";
exit;
```

Output :

```
Here is the starting DNA:
ACGGGAGGACGGGAAAATTACTACGGCATTAGC
Here is the reverse complement DNA:
GGAAAAGGGGAAGAAAAAAAGGGGAGGAGGGGA
That was a bad algorithm, and the reverse complement was wrong!
Try again ...
Here is the reverse complement DNA:
GCTAATGCCGTAGTAATTTTCCCGTCCTCCCGT
This time it worked!
```

MODULE B: FILE I/O & CONTROL STATEMENT.

Exp 7 Reading protein sequence data from a file

Aim:
To Read a protein sequence data from a file using PERL

Program :

```
#!/usr/bin/perl -w
# Reading protein sequence data from a file, take 2
# The filename of the file containing the protein sequence
data
$proteinfilename = 'NM_021964fragment.pep';
# First we have to "open" the file, and associate
# a "filehandle" with it. We choose the filehandle
# PROTEINFILE for readability.
open(PROTEINFILE, $proteinfilename);
# Now we do the actual reading of the protein sequence data
from the file,
# by using the angle brackets < and > to get the input from
the
# filehandle. We store the data into our variable $protein.
#
# Since the file has three lines, and since the read only
is
```

```
# returning one line, we'll read a line and print it, three
times.
# First line
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the first line of the protein file:\n\n";
print $protein;
# Second line
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the second line of the protein file:\n\n";
print $protein;
# Third line
$protein = <PROTEINFILE>;
# Print the protein onto the screen
print "\nHere is the third line of the protein file:\n\n";
print $protein;
# Now that we've got our data, we can close the file.
close PROTEINFILE;
exit;
```

Output:

```
Here is the first line of the protein file:
MNIDDKLEGLFLKCGGIDEMQSSRTMVVMGGVSGQSTVSGELQD
Here is the second line of the protein file:
SVLQDRSMPHQEILAADEVLQESEMRQQDMISHDELMVHEETVKNDEEQMETHERLPQ
Here is the third line of the protein file:
GLQYALNVPISVKQEITFTDVSEQLMRDKKQIR
```

Exp 8 **Conditional Statement**

Aim:

To use the conditional statements in PERL

Program

```
#!/usr/bin/perl -w
# if-elsif-else
$word = 'MNIDDKL';
# if-elsif-else conditionals
if($word eq 'QSTVSGE') {
print "QSTVSGE\n";
} elsif($word eq 'MRQQDMISHDEL') {
print "MRQQDMISHDEL\n";
} elsif ( $word eq 'MNIDDKL' ) {
print "MNIDDKL--the magic word!\n";
} else {
print "Is \"$word\" a peptide? This program is not
sure.\n";
}
```

```
exit;
```

Output :
MNIDDKL--the magic word!

MODULE C: REGULAR EXPRESSIONS.

Exp 9 REGULAR EXPRESSIONS

Program 1

```
#!/usr/bin/perl
print “Enter your DNA Sequence : ”;
$DNA = <>;
chomp $DNA;
print “EcoRI site found!” if $DNA =~/GAATTC/;
print$DNA;
```

Program 2

```
#!/usr/bin/perl
$string = "do the words heaven and eleven match?";
if ( $string =~ /even/ )
{
print "A match was found.\n";
}
else
{
print "No match was found.\n";
}
```

Exp 10 Array operations

Aim:
To perform various array operations using PERL

Program 1:Pop operation using arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base1 = pop @bases;
print "Here's the element removed from the end: ";
print $base1, "\n\n";
print "Here's the remaining array of bases: ";
print "@bases";
```

OutPut :

```
Here's the element removed from the end: T
```

Here's the remaining array of bases: A C G

Program 2:Shift operation on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base2 = shift @bases;
print "Here's an element removed from the beginning: ";
print $base2, "\n\n";
print "Here's the remaining array of bases: ";
print "@bases";
```

Output :
Here's an element removed from the beginning: A
Here's the remaining array of bases: C G T

Program 3:Unshift operations on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base1 = pop @bases;
unshift (@bases, $base1);
print "Here's the element from the end put on the beginning:";
print "@bases\n\n";
```

Output:

Here's the element from the end put on the beginning: T A C G

Program 4:Push operation on arrays

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
$base2 = shift @bases;
push (@bases, $base2);
print "Here's the element from the beginning put on the end:";
print "@bases\n\n";
```

Output:

Here's the element from the beginning put on the end: C G T A

Program 5:Reverse of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
@reverse = reverse @bases;
print "Here's the array in reverse: ";
print "@reverse\n\n"
```

Output:

Here's the array in reverse: T G C A

Program 6:Length of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
print "Here's the length of the array: ";
print scalar @bases, "\n";
```

Output:

Here's the length of the array: 4

Program 7:Splicing of an array

```
#!/usr/bin/perl –w
@bases = ('A', 'C', 'G', 'T');
splice ( @bases, 2, 0, 'X');
print "Here's the array with an element inserted after the
2nd element: ";
print "@bases\n";
```

Output:

Here's the array with an element inserted after the 2nd element: A C X G T

MODULE D:SUBROUTINES & FUNCTIONS

Exp 11 Searching for motifs

Aim:
To search for a motif in a DNA sequence using PERL

Program :

```
#!/usr/bin/perl -w
# Searching for motifs
# Ask the user for the filename of the file containing
# the protein sequence data, and collect it from the
keyboard
print "Please type the filename of the protein sequence
data: ";
$proteinfilename = <STDIN>;
# Remove the newline from the protein filename
chomp $proteinfilename;
# open the file, or exit
unless ( open(PROTEINFILE, $proteinfilename) ) {
```

```
print "Cannot open file \"$proteinfilename\"\n\n";
exit;
}
# Read the protein sequence data from the file, and store
it
# into the array variable @protein
@protein = <PROTEINFILE>;
# Close the file - we've read all the data into @protein
now.
close PROTEINFILE;
# Put the protein sequence data into a single string, as
it's easier
# to search for a motif in a string than in an array of
# lines (what if the motif occurs over a line break?)
$protein = join( '', @protein);
# Remove whitespace
$protein =~ s/\s//g;
# In a loop, ask the user for a motif, search for the motif,
# and report if it was found.
# Exit if no motif is entered.
do {
print "Enter a motif to search for: ";
$motif = <STDIN>;
# Remove the newline at the end of $motif
chomp $motif;
# Look for the motif
if ( $protein =~ /$motif/ ) {
print "I found it!\n\n";
} else {
print "I couldn\'t find it.\n\n";
}
# exit on an empty user input
} until ( $motif =~ /^\s*$/ );
# exit the program
exit;
```

Output :

```
Please type the filename of the protein sequence data:
NM_021964fragment.pep
Enter a motif to search for: SVLQ
I found it!
Enter a motif to search for: jkl
I couldn't find it.
Enter a motif to search for: QDSV
I found it!
Enter a motif to search for: HERLPQGLQ
I found it!
Enter a motif to search for:
```

I couldn't find it.

Exp 12 A subroutine to append ACGT to DNA

Aim:
To append ACGT to DNA using subroutine

Program :

```
#!/usr/bin/perl -w
# A program with a subroutine to append ACGT to DNA
# The original DNA
$dna = 'CGACGTCTTCTCAGGCGA';
# The call to the subroutine "addACGT".
# The argument being passed in is $dna; the result is saved
in $longer_dna
$longer_dna = addACGT($dna);
print "I added ACGT to $dna and got $longer_dna\n\n";
exit;
# Here is the definition for subroutine "addACGT"
sub addACGT {
my($dna) = @_;
$dna .= 'ACGT';
return $dna;
}
```

Output:

```
I added ACGT to CGACGTCTTCTCAGGCGA and got
CGACGTCTTCTCAGGCGAACGT
```

Exp 13: Biological Databases-UniProt

Aim:

To retrieve the sequence of the Human keratin protein from UniProt database and to interpret the results.

Description:

The Universal Protein Resource (UniProt) is a comprehensive resource for protein sequence and annotation data. The UniProt databases are the UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), and the UniProt Archive (UniParc). The UniProt Metagenomic and Environmental Sequences (UniMES) database is a repository specifically developed for metagenomic and environmental data. UniProt is a collaboration between the European Bioinformatics Institute (EBI), the Swiss Institute of Bioinformatics (SIB) and the Protein Information Resource (PIR).

The UniProt Knowledgebase (UniProtKB) is the central hub for the collection of functional information on proteins, with accurate, consistent and rich annotation. In addition to capturing the core data mandatory for each UniProtKB entry (mainly, the amino acid sequence, protein name or description, taxonomic data and citation information), as much annotation information as possible is added. This includes widely accepted biological ontologies, classifications and cross-references, and clear indications of the quality of annotation in the form of evidence attribution of experimental and computational data.

The UniProt Knowledgebase consists of two sections: a section containing manually-annotated records with information extracted from literature and curator-evaluated computational analysis, and a section with computationally analyzed records that await full manual annotation. For the sake of continuity and name recognition, the two sections are referred to as "UniProtKB/Swiss-Prot" (reviewed, manually annotated) and "UniProtKB/TrEMBL" (unreviewed, automatically annotated), respectively.

Procedure:

1. Open the uniprot website
2. Type the protein name in the text box titled enter keyword
3. On pressing search button the result page is displayed
4. Choose the first sequence by double clicking the accession number
5. Take the fasta sequence.
6. Interpret the results.

Interpretation:

Results:

Exp 14 GenBank

Aim:

To retrieve the sequence of the Human keratin protein from GenBank database and to interpret the results.

Procedure:

The **GenBank** sequence database is an open access, annotated collection of all publicly available nucleotide sequences and their protein translations. This database is produced at National Center for Biotechnology Information (NCBI) as part of the International Nucleotide Sequence Database Collaboration, or INSDC. GenBank and its collaborators receive sequences produced in laboratories throughout the world from more than 100,000 distinct organisms. GenBank continues to grow at an exponential rate, doubling every 18 month. Release 155, produced in August 2006, contained over 65 billion nucleotide bases in more than 61 million sequences. GenBank is built by direct submissions from individual laboratories, as well as from bulk submissions from large-scale sequencing centers. Direct submissions are made to GenBank using BankIt, which is a Web-based form, or the stand-alone submission program, Sequin. Upon receipt of a sequence submission, the GenBank staff assigns an Accession number to the sequence and performs quality assurance checks. The submissions are then released to the public database, where the entries are retrievable by Entrez or downloadable by FTP. Bulk submissions of Expressed Sequence Tag (EST), Sequence-tagged site (STS), Genome Survey Sequence (GSS), and High-Throughput Genome Sequence (HTGS) data are most often submitted by large-scale sequencing centers. The GenBank direct submissions group also processes complete microbial genome sequences.

Procedure:

1. Open the genbank website
2. Type the protein name in the text box titled enter keyword
3. Select the appropriate database as nucleotide /protein
4. On pressing search button the result page is displayed
5. Choose the first sequence by double clicking the accession number
6. Take the fasta sequence.
7. Interpret the results.

Interpretation:

Results

Exp 15 PROTEIN DATA BANK

Aim:

To retrieve the structure of a protein and viewing it in RASMOL viewer.

Description:

The **Protein Data Bank** (**PDB**) is a repository for the 3-D structural data of large biological molecules, such as proteins and nucleic acids. The data, typically obtained by X-ray crystallography or NMR spectroscopy and submitted by biologists and biochemists from around the world, can be accessed at no charge on the internet. The PDB is overseen by an organization called the Worldwide Protein Data Bank.

The PDB is a key resource in areas of structural biology, such as structural genomics. Most major scientific journals, and some funding agencies, such as the NIH in the USA, now require scientists to submit their structure data to the PDB. If the contents of the PDB are thought of as primary data, then there are hundreds of derived (i.e., secondary) databases that categorize the data differently. For example, both SCOP and CATH categorize structures according to type of structure and assumed evolutionary relations; GO categorize structures based on genes.

Procedure:

1. Open the PDB website
2. Type the protein name in the text box titled enter keyword or type the PDB ID
3. On pressing search button the result page is displayed
4. Choose the appropriate structure by double clicking the PDB ID
5. A web page is displayed with details about the structure
6. Download the structure file from the right hand corner of the webpage
7. Save the file as PDB file.
8. Open the RASMOL viewer to view the downloaded structure.
9. Interpret the results.

Interpretation:

Results:

SEQUENCE ANALYSIS TOOLS

Exp 16 **BLAST**

Aim:

To find the similarity between sequences using BLAST

Description:

Basic Local Alignment Search Tool, or BLAST, is an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences. A *BLAST search* enables a researcher to compare a query sequence with a library or database of sequences, and identify library sequences that resemble the query sequence above a certain threshold. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm that seeks local as opposed to global alignments and is therefore able to detect relationships among sequences that share only isolated regions of similarity

There are many different types of BLAST available from the main BLAST web page. Choosing the right one depends on the type of sequence you are searching with (long, short; nucleotide protein), and the desired database. They are,

blastn -Nucleotide-nucleotide BLAST
blastp -Protein-protein BLAST
PSI-BLAST -Position-Specific Iterative BLAST
blastx -Nucleotide 6-frame translation-protein
tblastx -Nucleotide 6-frame translation-nucleotide 6-frame translation
tblastn -Protein-nucleotide 6-frame translation
megablast -Large numbers of query sequences

Procedure:

1. Open the Basic BLAST search page.

2. From the "Program" Pull Down Menu select the appropriate program.

3. Open your FASTA formatted sequence in a text editor as plain text.

4. Copy the entire sequence and paste it in the field titled "Enter your input data here", by clicking it once.

5. Set the pull down menu to "Sequence in FASTA format".

6. Make sure you have selected the correct BLAST program and BLAST database.

7. If you have entered your FASTA sequence or an Accession or GI number, click the "Submit Query Button".

8. BLAST will now open a new window and tell you it is working on your search.

9. Once your results are computed they will be presented in the window.

Interpretation

Results:

Exp 17 **FASTA**

Aim:

To find the similarity between sequences using **FASTA**

Description:

FASTA is a DNA and Protein sequence alignment software package first described by David J. Lipman and William R. Pearson in 1985 in the article Rapid and sensitive protein similarity searches. The original FASTP program was designed for protein sequence similarity searching. FASTA, described in 1988 added the ability to do DNA:DNA searches, translated protein:DNA searches, and also provided a more sophisticated shuffling program for evaluating statistical significance. There are several programs in this package that allow the alignment of protein sequences and DNA sequences. FASTA is pronounced "FAST-Aye", and stands for "FAST-All", because it works with any alphabet, an extension of "FAST-P" (protein) and "FAST-N" (nucleotide) alignment.

The current FASTA package contains programs for protein:protein, DNA:DNA, protein:translated DNA (with frameshifts), and ordered or unordered peptide searches. Recent versions of the FASTA package include special translated search algorithms that correctly handle frameshift errors (which six-frame-translated searches do not handle very well) when comparing nucleotide to protein sequence data.

In addition to rapid heuristic search methods, the FASTA package provides SSEARCH, an implementation of the optimal Smith-Waterman algorithm. A major focus of the package is the calculation of accurate similarity statistics, so that biologists can judge whether an alignment is likely to have occurred by chance, or whether it can be used to infer homology. The FASTA package is available from fasta.bioch.virginia.edu.

The web-interface to submit sequences for running a search of the European Bioinformatics Institute (EBI)'s online databases is also available using the FASTA programs.

Procedure:

1. Open the Basic FASTA search page.

2. From the "Program" Pull Down Menu select the appropriate program.The program is based on the query sequence.

3.Select the database against which the search has to be carried out.

4 From the results drop down box select an option.Enter your email id in the text box titled “Your email”

3. Open your FASTA formatted sequence in a text editor as plain text.

4. Copy the entire sequence and paste it in the field titled "Enter or paste sequence data", by clicking it once.

5. Set the pull down menu to "Sequence in FASTA format".

6. Make sure you have selected the correct FASTA program and database.

7. If you have entered your FASTA sequence or an Accession or GI number, click the "Submit Query Button".

8. FASTA will now open a new window and tell you it is working on your search.

9. Once your results are computed they will be presented in the window.

Interpretation:

Results:

Exp 18 **CLUSTALW**

Aim:

To align more than two sequences and find out the similarity between those sequences

Description:

Multiple alignments of protein sequences are important tools in studying sequences. The basic information they provide is identification of conserved sequence regions. This is very useful in designing experiments to test and modify the function of specific proteins, in predicting the function and structure of proteins, and in identifying new members of protein families.

Sequences can be aligned across their entire length (global alignment) or only in certain regions (local alignment). This is true for pairwise and multiple alignments. Global alignments need to use gaps (representing insertions/deletions) while local alignments can avoid them, aligning regions between gaps. ClustalW is a fully automatic program for global multiple alignment of DNA and protein sequences. The alignment is progressive and considers the sequence redundancy. Trees can also be calculated from multiple alignments. The program has some adjustable parameters with reasonable defaults.

Procedure:

1. Go to http://www.ebi.ac.uk/clustalw/

2. Put in your e-mail (if you want the results e-mailed to you not necessary) and an alignment title of your choice.

3. Paste sequences in the box below using the FASTA format:

>Name of Sequence #1

>Name of Sequence #2

4. Every sequence MUST have a different name given to it or the alignment will not work. Or Upload a file that includes all your sequences (such as a .doc file) in an acceptable format.

5. Press run button to start alignment reading.

6. When viewing your results, these are the consensus symbols used by ClustalW:

   a. "*" means that the residues or nucleotides in that column are identical in all sequences in the alignment.

   b. ":" means that conserved substitutions have been observed.

   c. "." means that semi-conserved substitutions are observed.

7. If you would like to see your results in color, push the button that displays Show Colors. Click Hide Colors to get rid of color..

8. Click on the button named View Alignment File to see the alignment on a larger scale (ie bigger font).

Interpretation:

Results:

Exp 19 EMBOSS

Aim :

To perform Sequence analysis by using EMBOSS.

Description :

EMBOSS (European Molecular Biology Open Software Suite) is an open source package of sequence analysis tools. This software covers a wide range of functionality and can handle data in a variety of formats. Extensive libraries are provided with the package, allowing users to develop and release their own software. EMBOSS also integrates a range of currently available packages and tools for sequence analysis, such as BLAST and ClustalW. A Java API (Jemboss) is also available.

EMBOSS contains around 150 programs (applications). These are just some of the areas covered:

- Sequence alignment.
- Rapid database searching with sequence patterns.
- Protein motif identification, including domain analysis.
- Nucleotide sequence pattern analysis, for example to identify CpG islands or repeats.
- Codon usage analysis for small genome.

Programs used for Nucleic acid sequence analysis :

1.Plot

Function : plot potential open reading frames in a nucleotide sequence

2. Restrict

Function :Report restriction enzyme cleavage sites in a nucleotide sequence

3. Transeq

Function :Translate nucleic acid sequences

4. Eprimer3

Function :Picks PCR primers and hybridization oligos

5. Backtranseq

Function :Back-translate a protein sequence to a nucleotide sequence

6. Dan

Function :Calculates nucleic acid melting temperature

7. Palindrome

Function :Finds inverted repeats in nucleotide sequence(s)

8. Revseq

Function :Reverse and complement a nucleotide sequence

Exp 20 PHYLIP

Aim :

To study the phylogentic relationships of nucleotide and protein sequence(s) by using PHYLIP Package.

Description:

PHYLIP (for **phyl**ogeny **i**nference **p**ackage) is a package now consisting of about 30 programs that cover most aspects of phylogenetic analysis. PHYLIP is free and available for a wide variety of computer platforms (Mac, DOS, UNIX, VAX/VMS,and others).

PHYLIP is a command-line program and does not have a point-and-click interface.The documentation is well written and very comprehensive,and the interface is straightforward.A program within PHYLIP is invoked by typing its name, which automatically causes the data to be read from a file called ‘‘infile’’ or a file name you specify if no infile exists.This infile must be in PHYLIP format; this format is clearly described in the documentation, and most sequence analysis programs offer the ability to export sequences in this format.For example, if an alignment is produced using CLUSTAL W or edited using GeneDoc,the alignment may be saved in PHYLIP format and then used in PHYLIP programs

directly. Once the user activates a given PHYLIP program and loads the infile, the user can then choose from an option menu or accept the default values.The program will write its output to a file called ‘‘outfile’’ (and ‘‘treefile’’ where applicable). If the output is to be read by another program, ‘‘outfile’’ or ‘‘treefile’’ must be renamed before execution of the next program, as all files named outfile/tree file in the current directory are overwritten at the beginning of any program execution.The tree file generated is a widely used format that can be imported into a variety of tree-drawing programs, including DRAWGRAM and DRAWTREE that come with this package.However, these PHYLIP tree-drawing programs produce low-resolution graphics, so a program such as TreeView (described below) is instead recommended. Particulars of some of the PHYLIP tree-inference programs are discussed below

Procedure :

1.Retrieve the nucleotide sequences form the genbank entrez page.
2.allign these sequences by using clustalw.
3.save the multiple sequence alignment in phylip format.
4.the aligned sequences will act as a input to the ‘phylip’program
5.selcet the phylip program from the phylip folder just by clicking the ‘dnadist’program.
6.the ‘dnadist’ program have options for selecting the distance parameters, click on of them ,it produces the distance table for the given sequences.

Exp 21 SWISS MODEL:

Aim:

To model a protein sequence using swiss model.

Description:

SWISS-MODEL (http://swissmodel.expasy.org) is a server for automated comparative modeling of three-dimensional (3D) protein structures. It pioneered the field of automated modeling starting in 1993 and is the most widely-used free web-based automated modeling facility today. In 2002 the server computed 120 000 user requests for 3D protein models. SWISS-MODEL provides several levels of user interaction through its World Wide Web interface: in the 'first approach mode' only an amino acid sequence of a protein is submitted to build a 3D model. Template selection, alignment and model building are done completely automated by the server. In the 'alignment mode', the modeling process is based on a user-

defined target-template alignment. Complex modeling tasks can be handled with the 'project mode' using DeepView (Swiss-PdbViewer), an integrated sequence-to-structure workbench. All models are sent back via email with a detailed modeling report. WhatCheck analyses and ANOLEA evaluations are provided optionally. The reliability of SWISS-MODEL is continuously evaluated in the EVA-CM project. The SWISS-MODEL server is under constant development to improve the successful implementation of expert knowledge into an easy-to-use server.

Procedure:

1. Go to NCBI (http://www.ncbi.nlm.nih.gov/) click on PUBMED activate a "protein" search in the menu and search for a particular protein.
2. Follow links to the amino acids sequence and copy that sequence to the computer's clipboard for use in the program BLAST. (you might also want to save it to a file as you will need it later).
3. Go back to the NCBI home page again and follow the "BLAST" link submit the sequence to BLAST selecting the PDB database alone. Click on the blue format button to see the results.
4. The resulting sequences are from sequences deposited with a known 3D structure and the four digit PDBcode is next to the words "pdb".
5. Record the PDBcodes for the different known 3D structures which align with the sequence.
6. Go to the PDB. http://www.rcsb.org/pdb/ and either type in a four digit code for one (or more) of the structures OR – use the "searchlight" functionality and search for "hslV" for example to see many related files at once.
7. Check the different PDBcodes to find the one which structure was solved to the highest resolution.
8. Download one or more of these structures from the PDB. The files may have an ".ent" file designation. These are equivalent to ".pdb" files
9. Run PYMOL or RASMOL and view your protein.
10. Now that you know that a reasonable structure exists, submit the sequence to the Swiss-Model web site. http://www.expasy.ch/swissmod/SWISS-MODEL.html Submit the original sequence with your e-mail address. The SWISS-MODEL server may take ~0.5-3 hours to return the results of the modeling exercise. You don't have to submit a PDB for SWISS-MODEL to use, it will use a mixture of all the top hits.
11. Receive several e-mails from SWISS-MODEL containing some introductory messages and the results of the modeling and PHD exercise.
12. Save the models to a file and view with PYMOL.

Interpretation:

Results:

Exp 22 **MODELLER**

Aim:

To model a protein sequence using modeler software.

Description:

MODELLER is a computer program that models three-dimensional structures of proteins and their assemblies by satisfaction of spatial restraints. MODELLER is most frequently used for homology or comparative protein structure modeling: The user provides an alignment of a sequence to be modeled with known related structures and MODELLER will automatically calculate a model with all non-hydrogen atoms.

More generally, the input to the program are restraints on the spatial structure of the amino acid sequence(s) and ligands to be modeled. The output is a 3D structure that satisfies these restraints as well as possible. Restraints can in principle be derived from a number of different sources. These include related protein structures (comparative modeling), NMR experiments (NMR refinement), rules of secondary structure packing (combinatorial modeling), cross-linking experiments, fluorescence spectroscopy, image reconstruction in electron microscopy, site-directed mutagenesis, intuition, residue-residue and atom-atom potentials of mean force, *etc*. The restraints can operate on distances, angles, dihedral angles, pairs of dihedral angles and some other spatial features defined by atoms or pseudo atoms. Presently, MODELLER automatically derives the restraints only from the known related structures and their alignment with the target sequence.

A 3D model is obtained by optimization of a molecular probability density function (pdf). The molecular pdf for comparative modeling is optimized with the variable target function procedure in Cartesian space that employs methods of conjugate gradients and molecular dynamics with simulated annealing.

MODELLER can also perform multiple comparison of protein sequences and/or structures, clustering of proteins, and searching of sequence databases. The program is used with a scripting language and does not include any graphics. It is written in standard FORTRAN 90 and will run on UNIX, Windows, or Mac computers.

1. Searching for structures related to target sequence.First, it is necessary to put the target sequence into the PIR format readable by MODELLER

2. A search for potentially related sequences of known structure can be performed by the **profile.build()** command of MODELLER. The following script, taken line by line, does the following (see file "build_profile.py"):
3. Initializes the 'environment' for this modeling run, by creating a new 'environ' object. Almost all MODELLER scripts require this step, as the new object (which we call here 'env', but you can call it anything you like) is needed to build most other useful objects.
4. Selecting a template.The output of the "build_profile.py" script is written to the "build_profile.log" file. MODELLER always produces a log file. Errors and warnings in log files can be found by searching for the "_*E*>" and "_*W*>" strings, respectively. MODELLER also writes the profile in text format to the "build_profile.prf" file. 3. Aligning TvLDF with the template
5. Model buildingOnce a target-template alignment is constructed, MODELLER calculates a 3D model of the target completely automatically, using its **automodel** class.
6. Model evaluation