ICB Fall 2009

# G4120: Introduction to Computational Biology

Oliver Jovanovic, Ph.D.
Columbia University
Department of
Microbiology & Immunology

Lecture 7
Internet Resources, Databases & Software
November 5, 2009

# Internet Protocols

**TCP/IP (Transmission Control Protocol/Internet Protocol)**
A multilayered protocol architecture that allows for the transmission of data over networks.

**TCP (Transmission Control Protocol)**
Provides reliable delivery of datagrams using connection oriented streaming with error detection and correction. Datagrams are packets of data containing the IP addressing information needed to switch them from one network to another until they arrive at their final destination.

**UDP (User Datagram Protocol)**
Provides low overhead connectionless datagram delivery.

**DHCP (Dynamic Host Configuration Protocol)**
Automates the configuration of computers that use TCP/IP by automatically assigning Internet addressing information from a network DHCP server.

**HTTP (HyperText Transfer Protocol)**
The protocol behind the World Wide Web (WWW). Recommend Apple's Safari or Mozilla's Firefox for browsing, **www.google.com** for searching.

**URL (Uniform Resource Locator)**
Used to locate resources on the WWW. Can have various prefixes, such as **http://www.columbia.edu** or **ftp://ftp.ncbi.nlm.nih.gov** or **file://Macintosh%20HD/Documents/Read%20Me.pdf** (note, a blank space in a URL " " is replaced by a "**%20**")

**NNTP (Network News Transfer Protocol)**
The newsgroup protocol used by Usenet, a world-wide distributed discussion system organized into newsgroups (e.g., **bionet.biology.computational, sci.bio.evolution** or **sci.med.informatics**). Recommend using **groups.google.com** to access Usenet.

# File Transfer and Telecommunications Protocols

**FTP (File Transfer Protocol)**
Allows you to transfer files to or from a remote machine running FTP. Usually anonymous, but insecure. Recommend using Fetch or Safari. Fetch is available free to students at:
**http://fetchsoftworks.com/Licensing/edustore.application.html**

**SFTP (Secure File Transfer Protocol)**
Allows you to securely transfer files with the data encrypted. Recommend using Fugu, free at:
**http://www.columbia.edu/acis/software/fugu/**

**AFP (Apple File Protocol)**
Allows you to transfer files to or from a Macintosh.

**SMB (Server Message Block)**
Allows you to transfer files to or from a PC.

**SMTP (Simple Mail Transfer Protocol)**
Used by email. Apple's Mail does a good job of preserving Macintosh file structure when sending attached files without requiring an additional compression step.

**SSH (Secure Shell)**
Can use to securely login remotely, or securely forward outgoing email.

**WebDAV (Web-based Distributed Authoring and Versioning)**
Allows you to collaborate with others on remote editing of documents. Used by iCal and iPhoto.

**AppleTalk and Bonjour**
Macintosh specific telecommunications protocols. AppleTalk is an older telecommunications protocol that still sees use, Bonjour can automatically discover available network resources.

# Bioinformatics and Computational Biology Internet Resources

**National Center for Biotechnology Information (NCBI)**

- PubMed, PubMed Central, Books and other reference material
- GenBank, RefSeq, CDD, MMDB and other sequence and structure databases
- Prokaryotic genome data and browsers (over 950 microbial, 3,500 virus and 1,900 plasmid)
- Eukaryotic genome data and browsers (over 400 genomes in progress, maps, partial sequences)
- BLAST, PSI-BLAST and VAST search tools, Cn3D visualization tool

**http://www.ncbi.nlm.nih.gov/**

**Ensembl (EMBL-EBI/Sanger Institute)**

Eukaryotic genome data and browsers, including primates (human, chimp, etc.), rodents (mouse, rat, etc.) various *Laurasiatheria* and *Afrotheria,* armadillo, marsupials, birds, reptiles, fish and other chordates and eukaryotes (*C. elegans, D. melanogaster,* and *S. cerevisiae* among others), database search and DNA and protein sequence analysis tools.

**http://www.ensembl.org/** and **http://www.ebi.ac.uk/**

**UCSC Genome Bioinformatics**

Genome data and browsers including vertebrates (human, chimp, rhesus, dog, cat, horse, cow, mouse, rat, opossum, chicken, *X. tropicalis,* zebrafish, *Tetraodon*, and fugu among others), deutrosomes (*C. intestinalis* and *S. purpuratus)* insects (*D. melanogaster, D. simulans* and *A. gambiae* among others*),* Nematodes (*C. elegans* and *C. briggsae* among others*);* and *S. cerevisiae* and SARS.

**http://genome.ucsc.edu/**

**Protein Data Bank (PDB) and Expert Protein Analysis System (Expasy)**

Worldwide repository for 3D protein structure data and biochemical information with links to useful tools, software and references for protein analysis.

**http://www.rcsb.org/pdb/** and **http://www.expasy.org/**

Lecture 7
Internet Resources, Databases & Software
November 5, 2009

# Bioinformatics and Computational Biology Software Resources

IU Bio-Archive (Macintosh, Unix and Java Molecular Biology Software)
**http://iubio.bio.indiana.edu/**

Pasteur Institute Bioinformatics Archive
**ftp://ftp.pasteur.fr/pub/GenSoft/**

European Bioinformatics Institute Biocatalog Software Directory
**http://www.biocatalogue.org**

Apple Computer Mac OS X Bioinformatics Tools
**http://www.apple.com/science/software/lifescience.html**

European Molecular Biology Open Software Suite (EMBOSS)
**http://emboss.sourceforge.net/**

TIGR/J. Craig Venter Institute Open Source Software Tools
**http://www.jcvi.org/cms/research/software/**

SourceForge, Fink and MacPorts Unix Ports to Mac OS X
**http://sourceforge.net**
**http://finkproject.org** and **http://pdb.finkproject.org/pdb/browse.php?=sci**
**http://www.macports.org/**

VersionTracker
**http://www.versiontracker.com**

# Databases

## Flat File Database (FFDB)

A collection of similar files made useful by ordering and indexing. All the information about one sequence would be stored in one structured text file, and you generally examine one file at a time.

**Examples:** GenBank, FileMaker (older versions)

## Relational Database (RDB)

All data is stored inside one or more tables of rows and column, with all operations done on the tables themselves or producing other tables as the result. All the information about one sequence would be stored in a collection of tables with other data, so you can easily look at just the information relating to that sequence, or how it relates to the database as a whole. Structured Query Language (SQL) is used to access data in a relational database.

**Examples:** MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle

## Object Oriented Databases (OODB)

Data is stored and retrieved in an fashion consistent with object oriented programming principles (based on languages such as Smalltalk, C++ or Java). They generally handle complex structures and concurrent interaction by multiple clients well. Many relational databases have or are acquiring object oriented database features.

**Examples:** PDB, Versant VDB, Gemstone GemFire

# Searching Sequence Databases

## Needleman-Wunsch

Needleman-Wunsch gives you the optimal global alignment of two sequences. This is best for comparing closely related sequences of similar lengths.

**Examples:** GCG Gap, EMBOSS Needle

## Smith-Waterman

Smith-Waterman gives you the optimal local alignment of two sequences. This is better for comparing distantly related sequences (where non-functional regions may have diverged).

**Examples:** GCG BestFit, EMBOSS Water

## BLAST

BLAST gives a fast approximation of Smith-Waterman, from 100 -1,000 times faster, but will not necessarily find optimal local alignments.

**Examples:** NCBI BLAST, WU-BLAST

# Rules of Thumb for BLAST

- The shortest possible word size (2 for proteins, 7 for nucleotides) gives the most sensitivity, though the search may take more time.

  ***Note:*** *A larger word size (3 for proteins, 11 for nucleotides) is the default setting for NCBI BLAST. You will have to change it manually.*

- At least initially, run your search with the Low Complexity filter off. Then, if you appear to be getting spurious hits, or for comparison purpose, run it again with the filter on. Although it can be helpful, the filter can also filter out a significant match.

  ***Note:*** *Filter on the default setting for NCBI BLAST. You will have to turn it off manually.*

- Keep in mind that BLAST is a heuristic version of Smith-Waterman, and may miss a significant alignment.

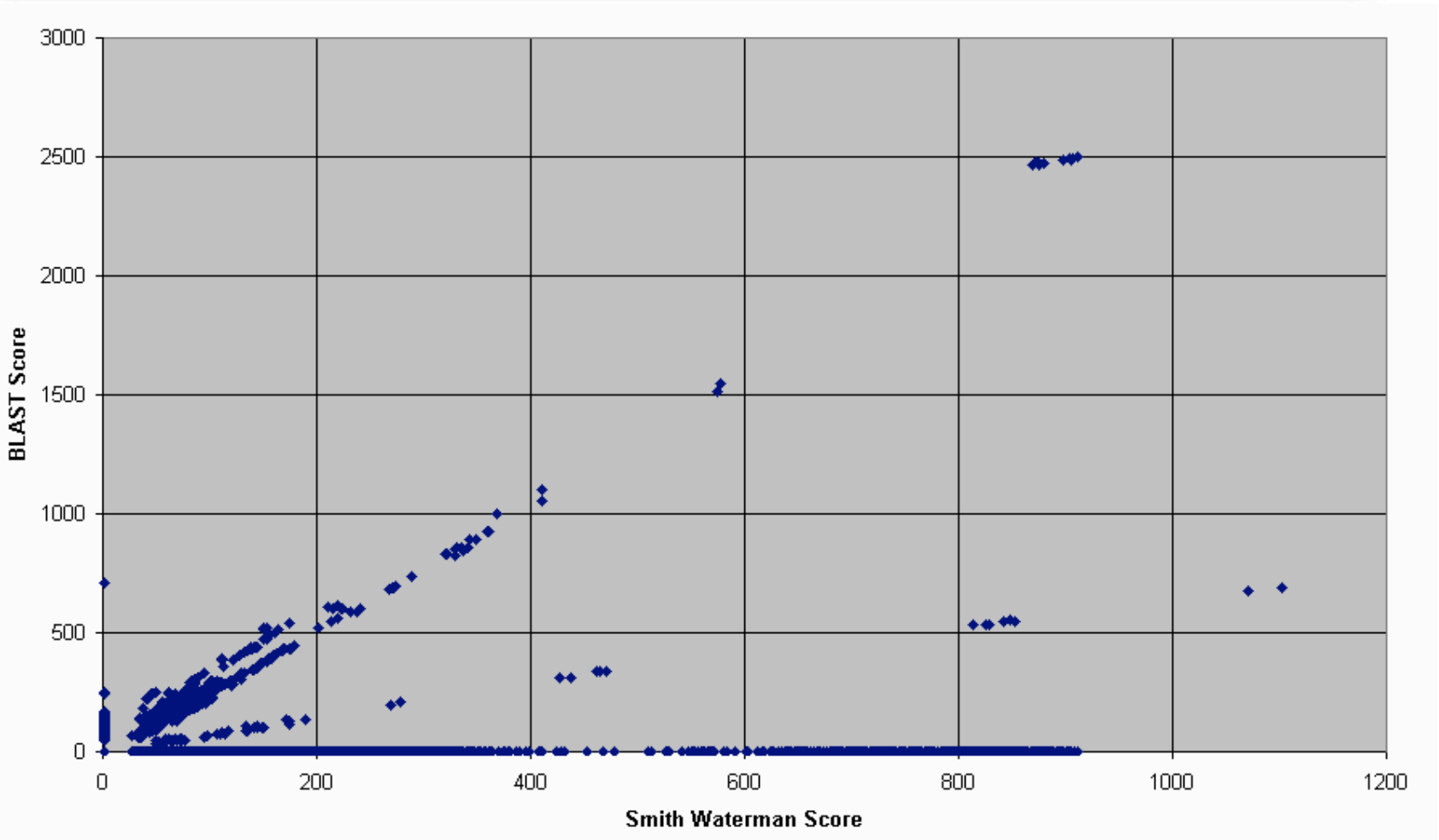
- The default BLOSUM62 substitution scoring matrix is best for comparing moderately distant and relatively closely related proteins. When searching for distantly related proteins, also try the PAM70 and BLOSUM45 (and PAM250) matrices. If comparing closely related proteins, also try the PAM30 and BLOSUM80 (and PAM1) matrices.

- PSI-BLAST can be useful for searching for very weak protein homologies.

- If searching with short DNA or protein sequences make sure you use the appropriate "Search for short nearly exact matches" BLAST page, or make sure to use those settings. BLAST is not the best tool to use for very short sequences.

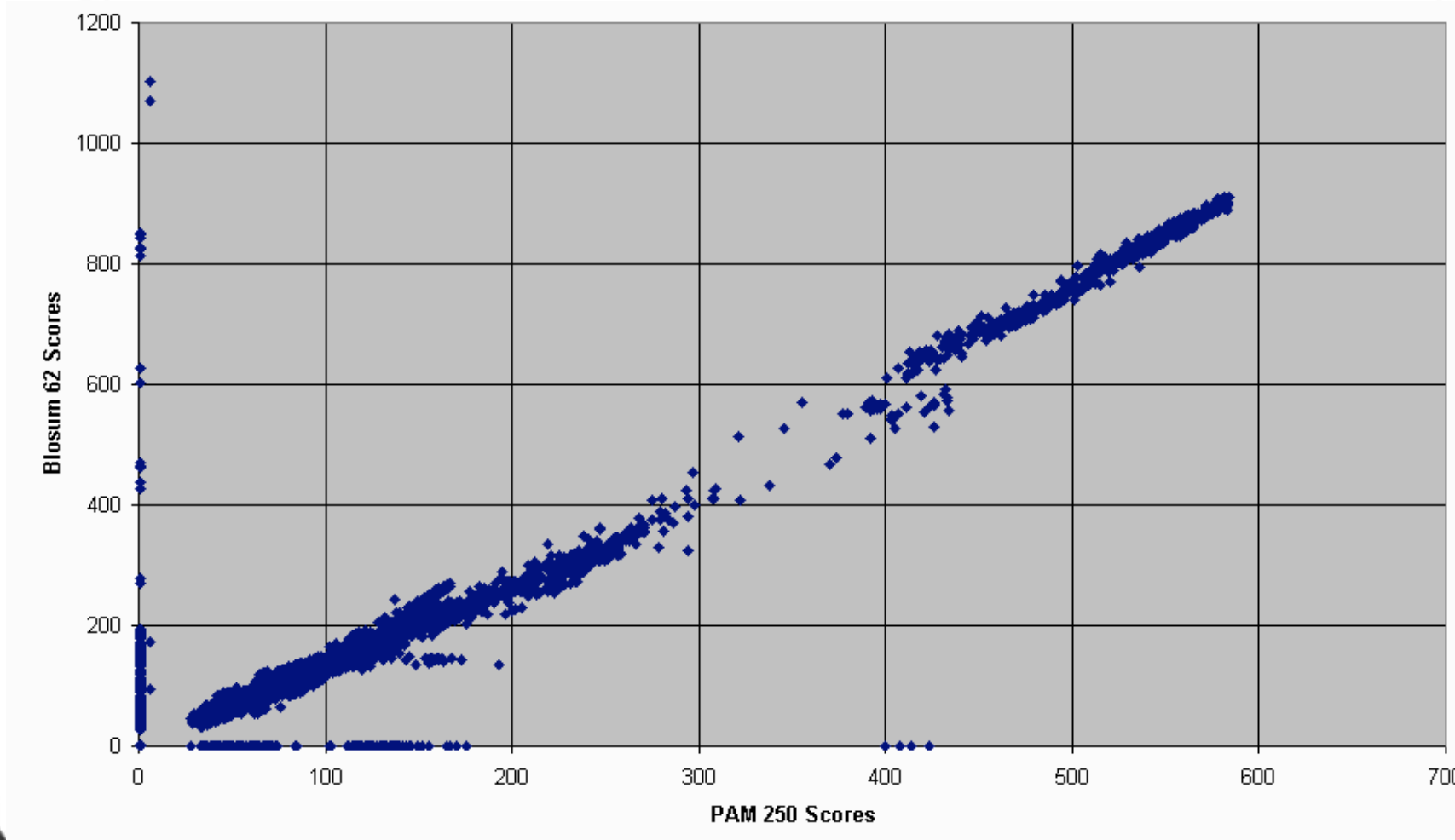
- The "Limit by entrez query" option allows BLAST searches to be limited to the results of an Entrez query against the database chosen, typically one or more organisms. Common organisms are provided in a popup menu. This can yield more relevant results.

# BLAST vs. Smith-Waterman




Lecture 7
Internet Resources, Databases & Software
November 5, 2009

# Blosum 62 vs. PAM 250



# Bioinformatics Programming

In addition to using software or websites created by others to perform a particular bioinformatics analysis, it is possible to write your own software programs, or modify existing programs to perform the analysis you need.

A wide variety of programming languages that are well suited to developing bioinformatics applications exist, from simpler languages such as Python, Ruby and Perl, to more complex languages such as C, C++, Cocoa and Java.

The Open Source software movement makes it particularly easy to adapt an existing program to your particular needs, or rapidly assemble a complex program from a number of free software programs. Open Source software is software distributed under an Open Source license that makes it available for free, along with the source code for the software.

Many Open Source libraries of bioinformatics tools and software exist for a variety of programming languages, including BioJava, BioPython, BioRuby, BioCocoa and BioPerl. These libraries provide modules and source code that can greatly simplify the development of bioinformatics applications.

Open Source software that has been widely adopted for general use includes the BSD Unix, Linux, FreeBSD and Apple Darwin operating systems, the Apache web server, the Netscape browser and the Java and Perl programming languages. Open Source tends to promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. Grant agencies are beginning to specify that software developed with public funds should be distributed under Open Source licenses. Thus, Open Source software is a particularly good choice for bioinformatics applications.

# Open Source Software

The Open Source Initiative (OSI) certifies Open Source licenses. To be OSI certified, the software must be distributed under a license that guarantees the right to read, redistribute, modify, and use the software freely. A variety of Open Source licenses exist, from the more permissive BSD style license (allows commercial resale) to the stricter GNU General Public License (GPL) license (any piece of software created with the code must remain free).

**Open Source Definition for Open Source Initiative Certification**

1. Free Redistribution May Not Be Restricted
2. Source Code in Unobfuscated Form Must Be Included or Made Readily Available
3. Derived Works and Modifications Must Be Allowed To Be Distributed Under the Same Terms
4. Integrity of The Author's Source Code Can Be Protected
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Specific Fields of Endeavor
7. License Applies to All to Whom the Program is Distributed
8. License Must Not Be Specific to a Product or Particular Software Distribution
9. License Must Not Restrict Other Software Distributed Along With It
10. License Must Be Technology and Interface Neutral

# Programming

## Programming
Programming involves giving a series of instructions to a computer that tell it to perform a task.

## Machine Code Programming
It is possible to program directly in the binary language of a computer (**0**'s and **1**'s). This is difficult and rarely done in modern programming.

## Programming Languages
Programming languages allow one to communicate with a computer using source code that is closer to a natural language, such as English. There are three main types of programming languages: (1) assembled, (2) interpreted and (3) compiled.

### Assembler
Automatically converts natural language into machine code. It is difficult and requires low level understanding of the machine, but can allow for the creation of highly optimized programs.

### Interpreter
An interpreter translates the source code written in a particular programming language into the appropriate machine code as the program is run. The translation is done dynamically. This type of program is often called a "script". Perl and Java are examples of interpreted languages (technically, their interpreters are interpreter/compliers). Many interpreted languages have an optional compiler.

### Compiler
A compiler compiles the source code written in a particular programming language into executable machine code, creating a separate executable program which will always run as machine code. C is an example of a compiled language. Many compiled languages offer an interpreter as well.

# Compiling a Simple C Program

**1)** Open a new document in TextWrangler and type the following source code:

```
#include <stdio.h>
main ()
{
        printf("Hello world?\n");
}
```

**2)** Save the file in your home directory as **hello.c** (don't append .txt).

**3)** The source code must then be compiled to run. We can use the gcc C compiler to do this by opening Terminal, then typing **gcc hello.c**

**4)** This creates a compiled executable program named **a.out** by default. Execute the newly created compiled program by typing **./a.out** or **~/a.out**

**5)** Note that in the C programming language, we have to first compile our source code to an executable program (the compiler automatically set the permissions of the **a.out** file to be executable), then run the compiled program. Other programming languages such as Perl are interpreted, which means that a text file containing source code for a Perl script can be directly executed (assuming the text file has permissions set to allow it to be executed).

# Compiling Unix Programs on OS X

## Downloading

The first step in running a Unix program is to download it, whether from a web site, FTP site, or email. Creating a specific directory to which to download Unix programs or move them to once they are downloaded, such as **~/bin**, is a good idea, and creating a specific directory for each program within **~/bin**, such as **~/bin/seqstat**, is an even better idea.

## Unarchiving

Most Unix programs come in a compressed or archived format. Stuffit Expander can often unarchive them properly, but sometimes this does not work properly. Fortunately, there are command line tools that will always work properly. Programs archived by Tar (**.tar**) format can be untarred from Terminal by issuing the command **tar -xvf *filename***. Tar files are often also compressed by **compress** (**.tar.z**) or **gzip** (**.tar.gz**). To simultaneously uncompress and untar a file, use the command **gnutar -xzvf *filename***. You can also do this in two steps, first uncompressing using **compress** or **gzip**, then untarring with **tar**.

## Compiling

Compiling a program involves running the appropriate compiler on a program's source code to convert it into an executable application. Typically, the compiler used will be gcc, the GNU project C and C++ compiler. One can use gcc directly to compile a simple program. More complex programs may first need to be configured with **configure**, then recompiled, tested and installed with **make**.

# Configure, Make and Install

To install and compile a complex Unix program, follow these steps:

**1)** Check for a file named **configure**. If one exists, run the command **./configure**. This will configure the installation for your system

**2)** Check for a file called **Makefile** or **make**. If it exists, run the command **make**. This will compile the program for your system. For simple programs, this may be the only step necessary

**3)** In some cases, you can test the compilation first by running the command **make test**

**4)** You may then need to finish the installation by running the command **make install**

If you have problems getting a Unix program to compile on OS X, often all that is needed is a minor change to the text in the **configure** or **make** file. Occasionally you may need to run an additional command such as **make install-lib** or **run build** instead. Sometimes the information can be found in the documentation or website for the program.

For more information on porting Unix programs to OS X, see:
**http://developer.apple.com/mac/library/documentation/Porting/Conceptual/PortingUnix/intro/intro.html**
and
**http://www.finkproject.org/doc/porting/index.php**

# Running Scripts on OS X

## Running Scripts

Running a script, such as a Perl script, simply involves downloading and uncompressing the script, making sure that the script has its permissions set to allow it to be executable, then running it. Some complex scripts may require that other scripts or libraries be installed before they can run.

## Permissions

To check permissions on a script, use the command **ls -l**. If the permissions list for the script does not have an **x** in it, is is not executable (typically it will start **-rwx**).

## Setting Permissions

To set the permissions of a script to be executable, use the program **chmod**. Issue the command **chmod 700 *filename*** to give only the owner (yourself) permission to execute. To give any user the ability to execute a script (but not change it) issue the command **chmod 755 *filename***.

## Run the Script

You can then execute the script by using the command ***./filename***. If you know the scripting language being used, you can call that directly, and pass a reference to the script name, e.g. issuing the command **perl *filename.pl***.

# Running a Script

## Running a Perl Script

**1)** Open a new document in TextWrangler and type the following source code:

```
#!/usr/bin/perl -w
use strict;
print "Hello, world?\n";
```

**2)** Save the file in your home directory as **hello.pl**

**3)** Try to run the script by typing **./hello.pl** and pressing Return

**4)** Make any necessary modifications

# Compiling a Program

## Compiling a Bioinformatics Program

**1)** Download seqstat.tar from **http://microbiology.columbia.edu/icb/seqstat.tar**

**2)** Create **~/bin**

**3)** Copy **seqstat.tar** to **bin**

**4)** In Terminal, type **cd ~/bin** and press Return, then type **tar -xvf seqstat.tar** and press Return

**5)** Type **cd seqstat** and press Return, then type **ls** and press Return to see what is there

**6)** Since a Makefile exists, simply type **make** and press Return to compile the program

**5)** To run the compiled program, type **./seqstat** and press Return. To quit a program while it is running, press **Control** and **C**

# References

## NCBI Field Guide and Exercises

**ftp://ftp.ncbi.nih.gov/pub/FieldGuide/Docs/handout.pdf**
**ftp://ftp.ncbi.nlm.nih.gov/pub/FieldGuide/NCBI_exercises.pdf**

## BLAST

*BLAST: An Essential Guide to the BASIC Local Alignment Search Tool* by Ian Korf, Mark Yandell & Joseph Bedell

## Databases

*Databases Demystified* by Andrew Oppel

*Database in Depth: Relational Theory for Practitioners* by C.J. Date

*MySQL, Fourth Edition* by Paul Dubois