Introduction to Bioinformatics

Prof. Dr. Nizamettin AYDIN
naydin@yildiz.edu.tr

Some useful bits & pieces that every Perl programmer should know

Strictness

• Perl breaks a number of the ‘‘golden rules’’ of the traditional programming language
  – allows variables to be used before they are declared
  – Subroutines can be invoked before they are defined
• All variables are global by default
  – the use of my variables turns a global variable into a lexical.
  – By default, the use of my variables is optional
    • However, it is possible to have perl insist on the use of my variables, making their use mandatory.

Strictness

• This insistence is referred to as strictness, and is switched on by adding the following line to the top of a program:
  
```
use strict;
```

• This is a directive that
  – tells perl to insist on all variables being declared before they are used,
  – all subroutines be declared (or defined) before they are invoked.

Strictness

• As programs get bigger, they become harder to maintain.
  – The use of use strict helps keep things organised and reduces the risk of errors being introduced into programs.
    • Anything that helps reduce errors is a good thing, even if it is sometimes inflexible.
  
• Thinking about the scope of variables, and using my and our to control the visibility of variables, becomes important as a program grows in size.

Results from bestrict

• When an attempt is made to execute the bestrict program, perl complains that the strictness rules have been broken:
  
```
>perl -w strict.pl
Global symbol "$message" requires explicit package name at bestrict line 7.
Global symbol "$message" requires explicit package name at bestrict line 9.
Execution of bestrict aborted due to compilation errors.
```

• These ‘‘compilation errors’’ are fixed by declaring the $message scalar as a my variable
  
```
my $message = "This is the message.\n";
```

• In the code, $message scalar is not declared as a lexical (my) or global (our) variable.
Results from bestrict

# bestrict - demonstrating the effect of strictness.
use strict;
my $message = "This is the message.\n";
print $message;

>perl -w mystrict.pl
This is the message.
>Exit code: 0

use subs

• Perl provides the use subs directive that ...
  – can be used in combination with use strict
    • to declare a list of subroutines at the top of the program

For example:
  use strict;
  use sub qw( drawline biod2mysql);

• The use subs directive declares a list of subroutine names that are later defined somewhere in the program’s disk-file.

Perl One-Liners

• Perl usually starts with the following line:
  >#!/ /usr/bin/perl –w
  w: warning
  – instructs perl to warn the programmer when it notices any dubious programming practices

• -e switch checks whether a module installed correctly or not:
  >perl -e 'use ExampleModule'
  e: execute
  – instructs perl to execute the program statements included within the single quotes

Unless you have a really good reason not to, always switch on strictness at the top of your program

Perl One-Liners

• Other examples:
  >perl -e 'print "Hello from a Perl one-liner.\n";'
  – a single line of Perl code is provided to perl to execute immediately from the command-line
  >perl -e 'print "$0.2f\n", 30000 * .12;'
  – turns perl into a simple command-line calculator
  The print subroutine is a variant of the more common print, and prints to a specified format.

• The ability to use the –e switch on the command-line in this way creates what is known in the perl world as...
  a one-liner.

Perl One-Liners: Equivalents

• Another useful switch is –n, which, when used with –e, treats the one-liner as if it is enclosed with a loop.

• Consider this one-liner:
  >perl -ne 'print if /ctgaatagcc/; embl.data
which is equivalent to the following program statement:

    while ( <> )
    {
      print if /ctgaatagcc/;
    }
Perl One-Liners: Equivalents

• When the one-liner is executed, the following output is generated:
  
  attgtaatat ctgaatagcc actgattttg taggcacctt tcagtccatc tagtgactaa

• Same function can also be implemented using grep:
  
  >grep 'ctgaatagcc' embl.data

• When the -n switch is combined with -p, the loop has a print statement added to the end.

Perl One-Liners: More Options

• Here is a one-liner that prints only those lines from the embl.data disk-file that do not end in four digits:
  
  >perl -ne 'last if /\d{4}$/;' embl.data

• The above one-liner is equivalent to this program:
  while ( <> )
    {
      last if /\d{4}$/;
    }
  continue {
    print $_;;
  }

• This one-liner is a little harder to do with grep.
  > grep -v '[0123456789][0123456789][0123456789][0123456789]$' embl.data

Perl One-Liners: More Options

• When executed, the following output is produced:
  
  gcacacagtc acagagagtc statatttaa accaaactct cttcctctga 60
  agacagagagtc staatttttag acattctctct ctctctctct 120
  tggacagagagtc gatattttag gcgggagta accctctctc 240
  agcagagagtc cagctctctct ggctctctct ctctctctct 360
  agcagagagtc cagctctctct gtctctctct ctctctctct 480
  tggacagagagtc gaatctctctct ggctctctct ctctctctct 600
  agcagagagtc cagctctctct ggctctctct ctctctctct 720
  agcagagagtc cagctctctct ggctctctct ctctctctct 840
  agcagagagtc cagctctctct ggctctctct ctctctctct 960
  
• The above one-liner is equivalent to this program:
  while ( <> )
    {
      last if /\d{4}$/;
    }
  continue {
    print $_;;
  }

• This one-liner is a little harder to do with grep.
  > grep -v '[0123456789][0123456789][0123456789][0123456789]$' embl.data

Running Other Programs From Perl

• There are two main ways to do this:
  
  – By invoking the program in such a way that after execution, the calling program can determine whether the called program successfully executed.
    • Perl’s in-built system subroutine behaves in this way
  
  – By invoking the program in such a way that after execution, any results from the called program are returned to the calling program.
    • Perl’s backticks and qx// operator behaves in this way

• Following example program demonstrates each of these mechanisms by invoking the DOS utility program, dir, that lists disk-files in the current directory

```perl
#!/usr/bin/perl –w
# pinvoke - demonstrating the invocation of other programs
# from Perl.
use strict;

my $result = system("dir *.*");
print "The result of the system call was as follows:\n".$result,"n";

$result = `dir *.*`;
# warning: $result = 'dir *.*'; will not work
print "The result of the backticks call was as follows:\n".$result,"n";

$result = qx/dir *./*/;
print "The result of the qx// call was as follows:\n".$result,"n";
```

```bash
#dir *./*
```
Running Other Programs From Perl

• The invocation of `system` results in the `dir` program executing. Any output from `dir` is displayed on screen (STDOUT) as normal:
  - As `dir` executed successfully, a value of zero is returned to `pinvoke` and assigned to the `$result` scalar.
  - If the `dir` program fails, the `$result` scalar is set to -1.

• Perl’s `backticks` (‘`‘ and ‘`‘) also execute external programs from within Perl:
  - The results from the program are captured and returned to the program.
  - In the `pinvoke` program, the results are assigned to the `$result` scalar and then printed to STDOUT as part of an appropriately worded message.

• The `qx//` operator is another way to invoke the `backticks` behaviour:
  - It works exactly the same way as `backticks`.

Recovering from Errors

• It is not always appropriate to `die` whenever an error occurs.
  - Sometimes it makes more sense to spot, and then recover from, an error.

• This is referred to as exception handling.

• Consider the following code:
  ```perl
  my $first_filename = "itdoesnotexist.txt";
  open FIRSTFILE, "$first_filename" or die "Could not open $first_filename. Aborting.\n";
  ```

Recovering from Errors

• Executing the code
  ```perl
  perl -w errorrec.pl
  ```
  Name "main::FIRSTFILE" used only once: possible typo at errorrec.pl line 4.
  Could not open itdoesnotexist.txt. Aborting.
  Exit code: 2

• This assumes that the `itdoesnotexist.txt` disk-file does not exist.
  - The program terminates as a result of the invocation of `die`.

• It is possible to protect this code by enclosing it within an `eval` block.
  ```perl
  eval {
    my $first_filename = "itdoesnotexist.txt"
    open FIRSTFILE, "$first_filename"
    or die "Could not open $first_filename. Aborting.\n";
  }
  ```

Recovering from Errors

• If `die` is invoked within an `eval` block, the block immediately terminates and Perl sets the internal `$@` variable to the message generated by `die`.

• After the `eval` block, it is a simple matter to check the status of `$@` and act appropriately.

• Adding the following if statement after the above `eval` block:
  ```perl
  if ( $@ ) {
    print "Calling eval produced this message: $\@\n";
  }
  ```
  prints the following message to STDOUT when the `itdoesnotexist.txt` disk-file does not exist:

  Calling eval produced this message: Could not open itdoesnotexist.txt. Aborting.
Use eval to protect potentially erroneous code

Sorting

• Perl provides powerful in-built support for sorting.
  • sort and reverse,
    • can be used to sort lists of strings or numbers into ascending order, descending order or any other customized order.
• Following examples demonstrate usage of sort and reverse.
  • In the following program a list of four short DNA sequences is assigned to an array called @sequences, which is then printed to STDOUT

```perl
#!/usr/bin/perl -w
# sortexamples - how Perl's in-built sort subroutine works.
use strict;
my @sequences = qw( gctacataat attgttttta aattatattc cgatgcttgg );
print "Before sorting:

- @sequences

my @sorted = sort @sequences;
my @reversed = sort { $b cmp $a } @sequences;
my @also_reversed = reverse sort @sequences;
print "Sorted order (default):

- @sorted

print "Reversed order (using sort { $b cmp $a }):

- @reversed

print "Reversed order (using reverse sort):

- @also_reversed

Results from sort examples ...

>perl -w sort1.pl
Before sorting:
  - gtacataat atgttttta aatatattc cgatgcttgg
Sorted order (default):
  - aatatattc cgatgcttgg gtacataat
Reversed order (using sort { $b cmp $a }):
  - gtacataat cgatgcttgg atgttttta aatatattc
Reversed order (using reverse sort):
  - gtacataat cgatgcttgg atgttttta aatatattc
>Exit code: 0
```

Another Sorting Example

• It is also possible to sort in numerical order using sort
  • The following program defines a list of chromosome pair numbers and assigns them to another array, called @chromosomes, and the array is then printed to STDOUT:

```perl
my @chromosomes = qw( 17 5 13 21 1 2 22 15 );
print "Before sorting/

- @chromosomes

my @sorted = sort { $a <=> $b } @chromosomes;
my @reversed = sort { $b <=> $a } @chromosomes;
print "Sorted order (using sort { $a <=> $b }):

- @sorted

print "Reversed order (using sort { $b <=> $a }):

- @reversed
```
And its results ...

```perl
> perl -w sort2.pl
Before sorting:
-> 17 5 13 21 1 2 22 15
Sorted order (using sort { $a <= $b }):
-> 1 2 5 13 15 17 21 22
Reversed order (using sort { $b <= $a }):
-> 22 21 17 15 13 5 2 1
> Exit code: 0
```

To learn more, use the following command-line to read the on-line documentation for `sort` that comes with Perl:

```bash
> perl -w sort
```

The sortfile Program

- The following program takes any disk-file and sorts the lines in the disk-file in ascending order:

```perl
#!/usr/bin/perl
#
# sortfile - sort the lines in any file.
# use strict;
my @the_file;
while ( <> )
  chomp;
  push @the_file, $_;
my @sorted_file = sort @the_file;
foreach my $line ( @sorted_file )
  print "$line
```

- Same thing could also be done by using Linux sort utility in command-line:

```bash
sort sort.data
```

Take the time to become familiar with the utilities included in the operating system

HERE Documents

- Consider the requirement to display the following text on screen in exactly the format shown from within a program:

```
Shotgun Sequencing
This is a relatively simple method of reading a genome sequence.
It is "simple" because it does away with the need to locate individual DNA fragments on a map before they are sequenced.
The Shotgun Sequencing method relies on powerful computers to assemble the finished sequence.
```

Without HERE Documents

- Could be done by using a sequence of print statements as follows:

```perl
print "Shotgun Sequencing\n
 print "This is a relatively simple method of reading a genome sequence. It is \"simple\" because it does away with the need to locate individual DNA fragments on a map before they are sequenced.\n```

The Shotgun Sequencing method relies on powerful computers to assemble the finished sequence.\n
print "sequence:\n```

Copyright 2000 N. AYDIN. All rights reserved.
Shotgun Sequencing

This is a relatively simple method of reading a genome sequence. It is ‘simple’ because it does away with the need to locate individual DNA fragments on a map before they are sequenced.

The Shotgun Sequencing method relies on powerful computers to assemble the finished sequence.

> Exit code: 0

With HERE Documents

• A better way to do this is to use Perl’s HERE document mechanism

```perl
my $shotgun_message = <<ENDSHOTMSG;
Shotgun Sequencing
This is a relatively simple method of reading a genome sequence. It is “simple” because it does away with the need to locate individual DNA fragments on a map before they are sequenced.
The Shotgun Sequencing method relies on powerful computers to assemble the finished sequence.
ENDSHOTMSG
print $shotgun_message;
```

> Exit code: 0

Even Better HERE Documents

• It is possible to improve previous program by removing the need for the Shotgun message scalar and printing the HERE document directly, as follows:

```perl
print <<ENDSHOTMSG;
Shotgun Sequencing
This is a relatively simple method of reading a genome sequence. It is “simple” because it does away with the need to locate individual DNA fragments on a map before they are sequenced.
The Shotgun Sequencing method relies on powerful computers to assemble the finished sequence.
ENDSHOTMSG
```

> Exit code: 0

HERE Documents

• HERE documents are useful,
  – especially when it comes to dynamically producing HTML documents.
    • This use of HERE documents will be discussed later
Downloading Datasets

• Downloading from the Web
  – a highly interactive mechanism
  – useful for downloading individual data-files
  – cumbersome for downloading large number of data-files

• Some technologies allow the easy integration of data sources across the Internet.
• However, it is often convenient to download frequently used datasets and store them locally.

Downloading Datasets

• The advantages of downloading and storing datasets locally:
  – Ease of access
    • accessing data-files on a local hard disk easier than writing an interface routine to download them as needed from a – possibly congested – location on the Internet.
  – Speed
    • Local hard-disk access, even over a shared file system, is usually faster than operating through external networks to Internet locations.
    – When the processing is performed locally, it may be possible to allocate extra computational resources to the analysis.

• The disadvantages of downloading and storing datasets locally:
  – Stale data
    • The local copy is a one-time “snapshot” of the dataset at a particular point in time.
    – At some stage, it will need to be updated or replaced by newer data.
  – Storage
    • The dataset has to be stored somewhere, and some datasets can be large.
    – The Protein Databank (PDB) is close to four gigabytes, and the PDB is one of the smaller databases!
    • Consequently, storing multiple copies of the PDB is often impractical.
  – Performance
    • The centralised specialist services accessible from the WWW are often configured with dedicated parallelised systems, designed to service requests as quickly as possible.
    – If the stored dataset is designed with such systems in mind, it is unlikely that a local system will be able to match this advanced processing capability.
    • Consequently, some analyses may be slower locally when compared to those performed on the WWW.

• can be accomplished in a number of ways:
  – by using established sequence analysis programs, such as EMBOSS
    http://emboss.sourceforge.net/
    • have specific methods for performing downloads.
    – Typically, datasets are accessed via a standard network connection to remote Internet sites.
    – Frequently, downloads are automated to occur at regular intervals.
  – The wget program, included with most Linux systems, can be used to do just this.
    • wget is an excellent example of GNU software as distributed by the Free Software Foundation.
      https://www.gnu.org/software/wget/
      http://gnuwin32.sourceforge.net/packages/wget.htm

Using wget to download PDB data-files

• To download a single data-file via anonymous FTP, simply provide the URL of the data-file required after the wget command.
  – For example, to download the two PDB structures, use these commands:
    mkdir structures
cd structures
wget ftp://ftp.rcsb.org/pub/pdb/data/structures/all/pdb1m7t.ent.Z
Mirroring a dataset

- **wget** can be used to mirror datasets.
  - To download the entire PDB, which is four gigabytes of data, stored in over 18000 data-files:
    - `wget --mirror ftp://ftp.rcsb.org/pub/pdb/data/structures/all/pdb`
  - Such a command should be invoked only when there is a real need to mirror the PDB.
  - A download of this size takes a considerable amount of time and disk space.
    - If such a need exists, once complete, another invocation of the same command downloads only additions or updates to the PDB since the last mirror.

Smarter mirroring

- The **wget** command slide 48 results in a deep directory tree.
  - The actual data-files are found in `structures/ftp.rcsb.org/pub/pdb/data/structures/all/pdb`.
  - Such a deep directory structure can be very inconvenient and frustrating to navigate.
  - Following **wget** invocation can help with this problem:
    - `wget --output-file log --mirror --http-user=anonymous --http-user-passwd=email@where.ever.net --directory-prefix=structures/mmCIF --no-host-directories --cut-dirs=6 ftp://ftp.rcsb.org/pub/pdb/data/structures/all/pdb`

Downloading a subset of a dataset

- On many occasions, the entire contents of an FTP site might not be required.
- **wget** can fetch a specific data-file, placing it in the current directory.
  - Use a command similar to this:
    - `wget ftp://beta.rcsb.org/pub/pdb/uniformity/data/mmCIF/all/1ger.cif.Z`
- While multiple URLs to data-files can be supplied on the command-line (separated by spaces), it is often more convenient to place the URLs in a data-file and use the "`--input file`" switch.

The **pdbselect** program takes the PDB-Select list produced in the Non-Redundant Datasets (discussed later), builds a list of URLs, removes the duplicates and then downloads them:

```perl
#! /usr/bin/perl
# pdbselect <list of PDB IDs> - a program that takes a list of PDB ID codes; build a list of URLs for them; and automates the downloading of them
# using 'wget'.
use strict;
use strict;
my $Base_URL = "ftp://ftp.rcsb.org/pub/pdb/data/structures/all/pdb";
my $Output_Dir = "structures";
open URL_LIST,">pdb_select_url.lst" or die "Cannot write to file: 'pdb_select_url.lst'"
while ( <>)
{
if ( /Failed/ )
{
next;
}
my ( $ID, $Length ) = split ( "\n", $_ );
my $Chain  = $Length;
my $ID =~ tr /[A-Z]/[a-z]/;
print URL_LIST "$Base_URL/pdb$ID.ent.Z\n";
}
```

Download a subset of a dataset

or die "Cannot write to file: 'pdb_select_url.lst'";
while ( <>)
{
if ( /Failed/ )
{
next;
}
s/ /\n/g;
my ( $Structure, $Length ) = split ( "\n", $_. );
my ( $SID, $Chain ) = split ( "", $Structure );
$SID =~ tr /[A-Z]/[a-z]/;
print URL_LIST "$Base_URL/pdb$SID.ent.Z\n";
}
**Downloading a subset of a dataset**

```bash
close URL_LIST;
if ( ! -e $Output_Dir )
{
    system "mkdir $Output_Dir";
}
if ( ! -w $Output_Dir or ! -d $Output_Dir )
{
    die "ERROR: Cannot access directory: '$Output_Dir'. Exiting
    in";
}
system "sort -u pdb_select_url.lst > unique_urls.lst";
system "rm $Output_Dir/* > /dev/null";
system "wget --output-file=log --http-user=anonymous
    --http-password=email@some.where.net
    --directory-prefix=$Output_Dir -i unique_urls.lst";
```

**The Protein Databank**

Download a dataset only when absolutely necessary. 
Consider the implications of doing so first.

**Determining Biomolecule Structures**

- There are many methods used for gaining information about the structure of a biomolecule.
- The two major methods by which the location of atoms can be determined to a useful accuracy:
  - X-Ray Crystallography
  - Nuclear Magnetic Resonance (NMR).

**The Protein Databank**

- Determining the detailed structure of a protein is more difficult than finding a DNA or amino acid sequence.
- The aim of some structural studies:
  - To know how the protein (or other biomolecule) "does what it does".
  - To alter its function.
    - For example, to design a small molecule that binds to the protein, more commonly known as a "drug".

- The similarity between the amino acid sequence of a "new" protein and one previously characterized can give an indication of the function of the new protein.
  - Sequence search algorithms assume some groups of amino acids have similar functional roles and consequently, occur in both sequences.
  - It is also assumed that these amino acids have similar local structures (the amino acids arrangement in space).
- It is these structures that determine the function of a protein.
  - Although these assumptions are are useful as a working model.

- Determining the detailed structure of a protein is more difficult than finding a DNA or amino acid sequence.
- The aim of some structural studies:
  - To know how the protein (or other biomolecule) "does what it does".
  - To alter its function.
    - For example, to design a small molecule that binds to the protein, more commonly known as a "drug".

- This program takes a list of PDB ID codes from STDIN and downloads them from the URL specified in the scalar variable $Base_URL6.
  - Those structures marked as Failed are skipped, otherwise a URL is built and written to the pdb_select_url.lst file.
  - Duplicate structures are filtered out using the sort -u operating system utility.
  - Error-checking is performed to see if the output directory exists (otherwise it is created) and that the directory can be accessed.
    - All previous files in it are then deleted using the rm system call.
  - Finally, wget is invoked with the list of URLs.
**X-Ray Crystallography**

- A technique for determining the three-dimensional structure of molecules,
  - including complex biological macromolecules such as proteins and nucleic acids.
- A powerful tool in the elucidation of the three-dimensional structure of a molecule at atomic resolution.
- Data is collected by diffracting X-rays from a single crystal, which has an ordered, regularly repeating arrangement of atoms.
- Based on the diffraction pattern obtained from X-ray scattering off the periodic assembly of molecules or atoms in the crystal, the electron density can be reconstructed.

**X-Ray Crystallography**

- A tool used for determining the atomic and molecular structure of a crystal.
- The underlying principle is that the crystalline atoms cause a beam of X-rays to diffract into many specific directions.
- By measuring the angles and intensities of these diffracted beams, a crystallographer can produce a 3D picture of the density of electrons within the crystal.

**X-Ray Crystallography**

- From this electron density image, the mean positions of the atoms in the crystal can be determined, as well as their chemical bonds, their disorder, and various other information.
  - The method revealed the structure and function of many biological molecules, including vitamins, drugs, proteins, and nucleic acids, such as DNA.
  - Note that the double helix structure of DNA discovered by James Watson and Francis Crick was revealed by X-ray crystallography.
- Recent advances in image reconstruction technology have made X-ray crystallography amenable to the structural analysis of much larger complexes, such as virus particles.

**X-Ray Crystallography**

- Viral capsid structure obtained by X-ray crystallography.
  - (A) Poliovirus capsid with T=3 symmetry.
  - (B) Hepatitis B virus capsid with T=4
- The major shortcomings of X-ray crystallography
  - it is difficult to obtain a crystal of virus particles, which is a prerequisite for X-ray crystallography.
  - X-ray crystallography generally requires placing the samples in nonphysiological environments, which can occasionally lead to functionally irrelevant conformational changes.

**Nuclear magnetic resonance**

- In NMR, no crystals are used in the process, and the protein remains in solution throughout the entire experiment.
- An intense and very linear magnetic field aligns the atomic nuclei of the protein into one of two spin states.
- A series of radio frequency pulses is used to perturb these by ‘‘flipping’’ some of the nuclei from one spin state to the other.
- As the total amount of energy absorbed is low, the protein remains undamaged and functions as normal.

**Nuclear magnetic resonance**

- Eventually, the ‘‘flipped’’ spin state of the nuclei realigns to the normal state, emitting a radio frequency pulse as it does so.
- The timing of this re-emission of energy is determined by the electronic environment in which the nucleus is embedded.
- A feature of this environment is the lectrostatic shielding effects of the surrounding nuclei.
- The nuclei, in addition to the bonds linking them, can be identified by their spin decay properties.
Nuclear magnetic resonance

- A problem with NMR methods is the size of the proteins that can be studied.
  - Using current techniques, this equates to a maximum of 200 amino acids.
  - This is low compared to the many hundreds of amino acids that can be studied using X-Ray Crystallography.
- The X-Ray Crystallography and NMR systems are complementary in many respects, as both determine, to a high accuracy, the coordinates of the atoms in protein structures.
  - If protein structures determined by X-Ray Crystallography and NMR are compared, they are generally consistent with each other and moreover are biologically plausible.
- This should give the researcher confidence when using them.

The Protein Databank

- contains a large collection of previously determined biological structures.
  - For inclusion in the PDB, the spatial locations of the atoms have to be determined with sufficient accuracy to usefully describe protein structures.
- also includes experimental details of how the structure was determined, what publications and other databases to consult for more information on the structure, some "derived data" and details of any ill-defined regions.
  - While this information is meant to be included in the PDB, some of it may be missing, incomplete or incorrect for some database entries.

One of the oldest bioscience data stores, dating back to 1971.
- It originally stored the 3D coordinates of protein structures as determined by the X-ray Crystallography method.
- Prior to the PDB, structures were typically published in journals, and many researchers re-entered the information manually into their computers so as to facilitate further manipulation of them.
- The original PDB data-file format adopted was a "flat" textual disk-file that was 80 columns wide.
- Today, the structures in the PDB are determined by either X-Ray Crystallography or NMR.
  - Often, many years of effort go into determining an individual structure.
- This is reflected in the growth of the number of entries in the PDB over some 40 years.

The PDB Data-file Formats

- available in one of two formats.
  - These formats are inter-convertible
- PDB flat file
  - The original, generic and highly unstructured PDB data-file format that is still widely used by researchers.
    - When biologists talk of "PDB files" or "PDB format", they are referring to this data-file format.
    - The current standard format is the 2.3 version.
- mmCIF
  - The new PDB data-file format that is designed to offer a highly structured, modern replacement to the original PDB Flat File format.
    - The mmCIF format is often informally referred to as the "new PDB format".

Example structures

- 1LQT
  - A modern, high-resolution "Oxidoreductase" enzyme structure produced using X-Ray Crystallographic techniques.
Example structures

- 1M7T
  - A modern protein structure of “Thioredoxin” produced using NMR.

Downloading PDB data-files

- PDB structure data-files can be downloaded from many web-site locations on the Internet.
  - The RCSB web-site is always a good place to start:
    - http://www.rcsb.org/pdb/
  - Alternatively, the EBI hosts a European mirror. Follow the links from:
    - http://www.ebi.ac.uk/services/ to access the PDB from the EBI.

Accessing Data in PDB Entries

- In a PDB data-file there is a left-right split (per line) and a top-bottom split (per data-file):
  - Left-right
    - The left-most characters (a maximum of nine) on each line indicate what information is present on the right-hand side.
  - Top-bottom
    - There is an upper HEADER section that contains the annotation about the structure (top) and a lower coordinates section that contains the 3D spatial locations of the atoms in the structure (bottom).

A short description of the most important fields in the PDB data-file

- HEADER
  - Contains a brief description of the structure, the date and the PDB ID code.
- TITLE
  - The title of the structure.
- COMPND
  - Brief details of the structure.
- SOURCE
  - Identifies which organism the structure came from.
- KEYWDS
  - Lists a set of useful words/phrases that describe the structure.
- AUTHOR
  - The scientists depositing the structure.
- REVDAT
  - The date of the last revision.

- JRNL
  - One or more literature references that describe the structure.
- REMARK 1 through REMARK 999
  - Details of the experimental methods used to determine the structure are contained in this subsection (see the example in the next section).
- DBREF
  - Cross links to other databases.
- SEQRES
  - The official amino acid sequence (protein, RNA or DNA) of the structure.
- HELIX/SHEET
  - Details of the regions of secondary structure found in the protein.
- ATOM/HETATM
  - The 3D spatial coordinates of particular atoms in the protein structure or other molecules such as water or co-factors.
Accessing PDB Annotation Data

- There are many examples of parsing data from the HEADER section of PDB data-files, all of which involve pattern matching.
  - Perl is exceptionally good at this.

  Two representative examples exploring
  - the relationship between the resolution of a structure and its Free R-value, both of which are measures of the quality of the X-Ray Crystallographic structures.
    - the Free R-value measures the agreement between the model and the observed x-ray reflection data.
    - The lower the Free R-value, the better the fit between the model and the observed data.
  - the database cross-referencing section used to link to other databases.

Free R and resolution

- The REMARK tag, type 2 subsection stores resolution, whereas the Free R-value is quoted in REMARK tag, type 3.
  - Here's a small extract from the 1LQT entry:

    REMARK 2
    REMARK 2 RESOLUTION 1.05 ANGSTROMS.

- In NMR structures, REMARK tag, type 2 and type 3 are present, but the data in them is "NOT APPLICABLE" for REMARK tag, type 2 and "NULL" or free text for REMARK tag, type 3.

- For the Free R-value:
  - The Free R-value measures the agreement between the model and the observed x-ray reflection data.
  - The lower the Free R-value, the better the fit between the model and the observed data.

Free R and resolution

- Structural Refinement is the process of iteratively fitting the model structure into the electron density map, and details of this refinement are stored in REMARK tag, type 3.

    REMARK 3 FIT TO DATA USED IN REFINEMENT.
    REMARK 3 CROSS-VALIDATION METHOD: THROUGHOUT.
    REMARK 3 FREE R VALUE (WORKING + TEST SET): 0.134
    REMARK 3 FREE R VALUE (TEST SET): 0.134
    REMARK 3 FREE R VALUE TEST SET SIZE (%): NULL
    REMARK 3 FREE R VALUE TEST SET COUNT: 2200

A Perl program extracts the resolution and Free R Value from any PDB data-files

```perl
#!/usr/bin/perl -w
# free_res - Designed to extract the 'Free R Value' and 'Resolution' quantities from 'PDB data-files' containing structures produced by 'Diffraction'.
use strict;
my $PDB_Path = shift;
open( INPUT_DIR, "$PDB_Path" ) or die "Error: Cannot read from PDB file: '$PDB_Path'
readdir( INPUT_DIR, @PDB_dir; close( INPUT_DIR; my @PDB_Files = grep /\.pdb/; @PDB_dir; foreach my $Current_PDB_File ( @PDB_Files )
  open( PDB_FILE, "$PDB_Path/$Current_PDB_File" ) or die "Cannot open PDB File: '$Current_PDB_File'
if ( $_ =~ /EXPDTA/ and !/DIFFRACTION/ )
  last;
if ( /^
REMARK 2 RESOLUTION/ )
  ( undef, undef, undef, $Resolution ) = split ( "\s+"; s_;
  if ( /^
REMARK 3 FREE R VALUE/ )
  $Free_R = substr ( $_; 47. 6); $Free_R =~ s/ //g;
```
Non-redundant Datasets

• There may be many reasons for redundancy in a dataset.
  – Scientific
    • It is often advantageous to study molecules with similar structures.
    • This is a classic scientific investigative methodology: change a small part, then identify the change in structure or function to form hypotheses about the reasons for the change.
    • Consequently, researchers are encouraged to study similar molecules to those studied previously.
  – Technological limitations
    • In X-Ray Crystallography, it is easier to obtain the structure of a molecule that is similar to one that is already known, as molecules with similar conformations are likely to have similar crystallisation conditions.
    • This, conveniently, allows two of the most difficult aspects of using X-Ray Crystallography to be dealt with.

Reduction of redundancy

• There are two reasons for supporting the reduction of a database:
  – Conceptually, to remove bias within the database.
    • The statistical analysis based upon the non-redundant dataset will be more representative of all the items in the database, rather than just the largest dominant group.
  – As a practical measure, to reduce the computational requirements caused by analysing examples that are unnecessary.
    • For example, the PDB-Select structural non-redundant dataset contains approximately 1600 protein structures, whereas the entire PDB contained approximately 18,000.

Database cross references

• The DBREF subsection gives a list of cross references to other Bioinformatics databases.
  – This makes it easier for researchers to integrate biological datasets.
  – The second value on the DBREF line is the PDB identifier.
  – By examining this value, researchers and automatic parsing programs can tell to which structure the entry belongs.
• Example DBREF lines:
  DBREF 1LQT A 1 456 GB 13882996 AAK47528 1 456
  DBREF 1LQT B 1 456 GB 13882996 AAK47528 1 456
  DBREF 1AFI 1 72 SWS P04129 MERP_SHIFL 20 91
  DBREF 1M7T A 1 66 SWS P10599 THIO_HUMAN 0 65
  DBREF 1M7T A 67 106 SWS P00274 THIO_ECOLI 68 107

Database cross references

• The PDB publishes a table of database names and their associated, abbreviated codes.

<table>
<thead>
<tr>
<th>Database Name</th>
<th>Database Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioMagResBank</td>
<td>BMRB</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>BLOCKS</td>
</tr>
<tr>
<td>European Molecular Biology Laboratory</td>
<td>EMBL</td>
</tr>
<tr>
<td>GenBank</td>
<td>GB</td>
</tr>
<tr>
<td>Genome Data Base</td>
<td>GD</td>
</tr>
<tr>
<td>Nucleic Acid Database</td>
<td>NDB</td>
</tr>
<tr>
<td>PROSITE</td>
<td>PROSITE</td>
</tr>
<tr>
<td>Protein Data Bank</td>
<td>PDB</td>
</tr>
<tr>
<td>Protein Identification Resource</td>
<td>PID</td>
</tr>
<tr>
<td>Swiss-Prot</td>
<td>SWISS-PROT</td>
</tr>
<tr>
<td>TREMBL</td>
<td>TREMBL</td>
</tr>
</tbody>
</table>

Coordinates section

• The coordinate data for the locations of atoms in the macromolecular structure is straightforward, especially when compared to the annotation contained in the HEADER section of the PDB data-file.
  – The coordinates are presented as points in space, the atoms they represent are actually in motion.
  – In crystallographic structures, isotropic B-factors, commonly referred to as "Temperature Factors", give an idea of the vibration of the molecule.
    • For very high-resolution structures, Anisotropic Temperature Factors may be included in the ANISOU lines.
      – These provide an idea of the vibration of the molecule in the directions of the coordinate axes.
  – In NMR structures, the variation in position of a particular atom between different models in the ensemble can be used as a similar measure of motion or as an indication of the error between the minimisation models.
• Here is an example from 1M7T:
  REMARK 210
  REMARK 210 BEST REPRESENTATIVE CONFORMER IN THIS ENSEMBLE: 21
  REMARK 210

Copyright 2000 N. AYDIN. All rights reserved.
Data section

• Referring to the 1LQT x-ray structure, an extract of lines from the coordinate section looks like this:

ATOM 1  N  ARG  A  2  26.318 -8.020 39.090 1.00 20.71  N
ANSEI 1  N  ARG  A  2  2040 3074 2775 1.14 -119 -391  N
ATOM 2  C  ARG  A  2  25.150 -8.702 38.565 1.00 18.45  C
ANSEI 2  CA  ARG  A  2  2029 2637 2415 0.67 -121 -106  C
ATOM 3  C  ARG  A  2  24.840 -8.176 37.183 1.00 17.33  C
ANSEI 3  CA  ARG  A  2  1689 2420 2829 0.43 -282 -258  C
ATOM 4  O  ARG  A  2  27.151 -7.648 30.775 1.00 18.14  O

Data section

• For the 1M7T NMR structure, an extract of lines from the coordinate section looks like this:

MODEL 1
ATOM 1  N  MET  A  1  3.110 -0.462 -3.055 1.00 0.00  N
ATOM 2  CA  MET  A  1  2.546 -3.712 -2.053 1.00 0.00  C
ATOM 3  C  MET  A  1  1.134 -3.295 -2.410 1.00 0.00  C
ATOM 4  O  MET  A  1  0.882 -2.130 -2.783 1.00 0.00  O
ATOM 5  C  MET  A  1  3.464 -2.091 -2.002 1.00 0.00  C
ATOM 6  C  MET  A  1  3.781 -1.003 -3.270 1.00 0.00  C
ATOM 7  C  N  MET  2  4.254 -0.166 -3.285 1.00 0.00  N
ATOM 8  C  N  MET  2  6.004 -0.207 -2.001 1.00 0.00  C
ATOM 9  C  N  MET  2  6.604 -0.207 -2.001 1.00 0.00  C
ATOM 10  N  MET  2  2.650 -0.801 -2.859 1.00 0.00  N
ATOM 11  H  MET  2  4.184 -0.801 -3.823 1.00 0.00  H
ATOM 12  H  MET  2  2.157 -4.178 -1.079 1.00 0.00  H
ATOM 13  H  MET  2  2.886 -1.718 -1.805 1.00 0.00  H
ATOM 14  H  MET  2  4.387 -2.778 -1.536 1.00 0.00  H
ATOM 15  H  MET  2  4.398 -2.461 -1.807 1.00 0.00  H
ATOM 16  H  MET  2  2.907 -1.939 -3.908 1.00 0.00  H
ATOM 17  H  MET  2  6.344 -3.302 -1.167 1.00 0.00  H
ATOM 18  H  MET  2  6.188 -0.120 -1.869 1.00 0.00  H
ATOM 19  H  MET  2  6.156 0.456 -3.105 1.00 0.00  H
ATOM 20  N  VAL  A  2  0.215 -4.256 -2.466 1.00 0.00  N

Data section

• In each ATOM line, the fields are as follows:

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>DATA TYPE</th>
<th>FIELD</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 6</td>
<td>Record name</td>
<td>XATOM(s)</td>
<td>Atom serial number, altserial number</td>
</tr>
<tr>
<td>7 - 11</td>
<td>Record name</td>
<td>AATOM(s)</td>
<td>Atom type from atom type dictionary</td>
</tr>
<tr>
<td>12 - 16</td>
<td>Record name</td>
<td>CHARAC</td>
<td>Character string: ( \text{Inhalt, \text{CHARAC}} )</td>
</tr>
<tr>
<td>17 - 20</td>
<td>Record name</td>
<td>ALTLOC</td>
<td>Alternate location indicator, code for insertion of coordinates, or atom type dictionary entry number</td>
</tr>
<tr>
<td>21 - 24</td>
<td>Record name</td>
<td>CHAREN</td>
<td>Character string: ( \text{ANEM, \text{CHAREN}} )</td>
</tr>
<tr>
<td>25 - 26</td>
<td>Record name</td>
<td>CHARG</td>
<td>Charge</td>
</tr>
<tr>
<td>27 - 30</td>
<td>Record name</td>
<td>ATOM</td>
<td>Code for insertion of residues, or atom type dictionary entry number, or atom type dictionary entry number for ( \text{ATMT} )</td>
</tr>
<tr>
<td>31 - 34</td>
<td>Record name</td>
<td>ORIGX</td>
<td>Original coordinate for ( \text{ZATOM} )</td>
</tr>
<tr>
<td>35 - 38</td>
<td>Record name</td>
<td>COORI</td>
<td>Orthogonal coordinates for ( \text{YATOM} )</td>
</tr>
<tr>
<td>39 - 42</td>
<td>Record name</td>
<td>RESI</td>
<td>Residue name</td>
</tr>
<tr>
<td>43 - 46</td>
<td>Record name</td>
<td>RESN</td>
<td>Residue sequence number</td>
</tr>
</tbody>
</table>

Extracting 3D co-ordinate data

• The technique involves extracting the three substrings from each line that contains the X, Y and Z coordinates.

• Assuming the data is in \( S_\_ \) three invocations of Perl’s substr subroutine do the trick:

```perl
my ($X, $Y, $Z) = ( substr($_, 30, 8), substr($_, 38, 8), substr($_, 46, 8) );
```
The simple_coord_extract program

```perl
#!/usr/bin/perl
#
# simple_coord_extract <PDB File> - Demonstrates the extraction of
# C-alpha coordinates from a PDB file.
#
use strict;
while ( <> ) {
    if ( /^ATOM/ && substr( $_, 13, 4 ) eq "CA " ) {
        my ( $X, $Y, $Z ) = ( substr( $_, 30, 8 ), substr( $_, 38, 8 ), substr( $_, 46, 8 ) );
        $X =~ s/ //g;
        $Y =~ s/ //g;
        $Z =~ s/ //g;
        print "X, Y & Z: $X, $Y, $Z \n";
    }
}
```

Results from simple_coord_extract ...

X, Y & Z: 25.150, -8.702, 38.505
X, Y & Z: 23.675, -8.497, 35.069
X, Y & Z: 20.747, -6.252, 34.332
X, Y & Z: 17.545, -8.297, 34.292
X, Y & Z: 15.182, -7.484, 31.454
X, Y & Z: 11.736, -8.952, 30.942
X, Y & Z: 10.261, -9.014, 27.451

Introducing Databases

- Many modern computer systems store vast amounts of structured data.
- Typically, this data is held in a database system.
- Database
  - a collection of one or more related tables.
- Table
  - a collection of one or more rows of data.
    - The rows of data are arranged in columns, with each intersection of a row and column containing a data item.
- Row
  - a collection of one or more data items, arranged in columns.
    - Within a row, the columns conform to a structure.

Introducing Databases

- For example,
  - if the first column in a row holds a date, then every first column in every row must also hold a date.
  - if the second column holds a name, then every second column must also hold a name, and so on.
- The following data corresponds to the structure, in that there are two columns, the first holding a date, the second holding a name:

<table>
<thead>
<tr>
<th>Discovery_Date</th>
<th>Scientist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960-12-21</td>
<td>P. Barry</td>
</tr>
<tr>
<td>1954-6-14</td>
<td>M. Moorhouse</td>
</tr>
</tbody>
</table>

Structured data

- Each column can be given a descriptive name.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery_Date</td>
<td>a valid Date</td>
</tr>
<tr>
<td>Scientist</td>
<td>a String no longer than 64 characters</td>
</tr>
</tbody>
</table>

- This type information generally goes by one of two names: metadata or schema.

Relating tables

- Extending the Discoveries table to include details of the discovery, an additional column is needed to hold the data:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery_Date</td>
<td>a valid Date</td>
</tr>
<tr>
<td>Scientist</td>
<td>a String no longer than 64 characters</td>
</tr>
<tr>
<td>Discovery</td>
<td>a String no longer than 128 characters</td>
</tr>
</tbody>
</table>
The problem with single-table databases

- Although the above table structure solves the problem of uniquely identifying each scientist, it introduces some other problems:
  - If a scientist is responsible for a large number of discoveries, their identification information has to be entered into every row of data that refers to them.
    - This is time-consuming and wasteful.
  - Every time identification information is added to a row for a particular scientist, it has to be entered in exactly the same way as the identification information added already.
    - Despite the best of efforts, this level of accuracy is often difficult to achieve.
  - If a scientist changes any identification information, every row in the table that refers to the scientist’s discoveries has to be changed.
    - This is drudgery.

Solving the one table problem

- The problems described in the previous section are solved by breaking the all-in-one Discoveries table into two tables.
- Here is a new structure for Discoveries:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery_Date</td>
<td>a valid Date</td>
</tr>
<tr>
<td>Scientist</td>
<td>a String no longer than 84 characters</td>
</tr>
<tr>
<td>Discovery</td>
<td>a String no longer than 128 characters</td>
</tr>
<tr>
<td>Date_of_birth</td>
<td>a valid Date</td>
</tr>
<tr>
<td>Telephone_number</td>
<td>a String no longer than 16 characters</td>
</tr>
</tbody>
</table>

Relational Databases

- Relating data in one table to that in another forms the basis of modern database theory.
  - It also explains why so many modern database technologies are referred to as Relational Database Management Systems (RDBMS).
- When a collection of tables is designed to relate to each other they are collectively referred to as a database.
- It is usually a requirement to give the database a descriptive name.

Database system: a definition

- A database system is a computer program (or group of programs)
  - that provides a mechanism to define and manipulate
    - one or more databases
- A database system
  - allows databases, tables and columns to be created and named, and structures to be defined.
  - provides mechanisms to add, remove, update and interact with the data in the database.
- Data stored in tables can be searched, sorted, sliced, diced and cross-referenced.
- Reports can be generated, and calculations can be performed.
Available Database Systems

- Personal database systems:
  - Designed to run on PCs
    - Access, Paradox, FileMaker, dBase
- Enterprise database systems:
  - Designed to support efficient storage and retrieval of vast amount of data
    - Interbase, Ingres, SQL Server, Informix, DB2, Oracle
- Open source database systems:
  - Free!!! (Linux!!!)
    - PostgreSQL, MySQL

Choosing Database System

- Which type of database system is chosen depends on a number of factors, including (but not limited to):
  - The amount of data to be stored in the database.
  - Whether the data supports a small personal project or a large collaborative one.
  - How much funds (if any) are available towards the purchase of a database system.

SQL: The Language of Databases

- Defining data with SQL (structured query language)
- SQL provides two facilities:
  - A database definition Language (DDL)
    - provides a mechanism whereby databases can be created
  - A Data Manipulation Language (DML)
    - provides a mechanism to work with data in tables

Installing a database system

- MySQL is a modern, capable and SQL-enabled database system.
- It is Open Source and freely available for download from the MySQL web-site: http://www.mysql.com
- It comes as a standard, installable component of most Linux distributions
- The following commands switch on MySQL on RedHat and RedHat-like Linux distributions
  - chkconfig --add mysqld
  - chkconfig mysqld on
- If the first chkconfig command produces an error messages like this:
  - error reading information on service mysqld: No such file or directory
  - this means that MySQL is not installed and the second command will also fail.

Installing a database system

- Once MySQL is installed, it needs to be configured.
- The first requirement is to assign a password to the MySQL superuser, known as “root”.
- The mysqladmin program does this, as follows:
  - mysqladmin -u root password 'passwordhere'
- It is now possible to securely access the MySQL Monitor command-line utility with the following command, providing the correct password when prompted:
  - mysql -u root -p

A Database Case Study: MER

- A small collection of SWISS-PROT and EMBL entries are taken from the Mer Operon, a bacterial gene cluster that is found in many bacteria for the detoxification of Mercury Hg2+ ions.
- These provide the raw data to a database, which is called MER.
- The MER database contains four tables:
  - proteins – A table of protein structure details, extracted from a collection of SWISS-PROT entries.
  - dnas – A table of DNA sequence details, extracted from a collection of EMBL entries.
  - crossrefs – A table that links the extracted protein structures to the extracted DNA sequences.
  - citations – A table of literature citations extracted from both the SWISS-PROT and EMBL DNA entries.
A Database Case Study: MER

• Once the raw data is in the database, SQL can be used to answer questions about the data.

• For instance:
  – How many protein structures in the database are longer than 200 amino acids in length?
  – How many DNA sequences in the database are longer than 4000 bases in length?
  – What’s the largest DNA sequence in the database?
  – Which protein structures are cross-referenced with which DNA sequences?
  – Which literature citations reference the results from the previous question?

Creating the MER database

• SQL queries can be entered directly at the MySQL Monitor prompt.
  mysql> create database MER;
  Query OK, 1 row affected (0.36 sec)

• A list of databases is returned by MySQL.
  • There are three identified databases:
    – MER – The just-created database that will store details on the extracted protein structures, DNA sequences, cross-references and literature citations.
    – test – A small test database that is used by MySQL and other technologies to test the integrity of the MySQL installation.
    – mysql – The database that stores the internal “system information” used by the MySQL database system.

Creating the MER database

• It is possible to use the MySQL superuser to create tables within the MER database.

• However, it is better practice to create a user within the database system to have authority over the database, and then perform all operations on the MER database as this user.

• The queries to do this are entered at the MySQL Monitor prompt.

• Here are the queries and the messages returned:
  mysql> use mysql;
  Database changed

  mysql> grant all on MER.* to bbp identified by 'passwordhere';
  Query OK. 0 rows affected (0.00 sec)

• The first query tells MySQL that any subsequent queries are to be applied to the named database, which in this case is the mysql database.

• The second query does three things:
  – creates a new MySQL user called “bbp”;
  – assigns a password with the value of “passwordhere” to user “bbp”;
  – grants every available privilege relating to the MER database to “bbp”.

Adding tables to the MER database

create table proteins
(
  accession_number varchar (6) not null,
  code varchar (4) not null,
  species varchar (5) not null,
  last_date date not null,
  description text not null,
  sequence_header varchar (75) not null,
  sequence_length int not null,
  sequence_data text not null
)

$ mysql -u bbp -p MER < create_proteins.sql

Understanding the data before designing the tables

Databases and Perl

• Why Program Databases?
  – Customised output handling
    • Programs can be written to post-process the results of any SQL query and display them in any number of preferred formats.
  – Customised input handling
    • Users of customised input handling programs do not need to know anything about SQL – all they need to know and understand is their data.
  – Extending SQL
    • Some tasks that are difficult or impossible to do with SQL can be programmed more easily.
  – Integrating MySQL into custom applications
    • Having the power of MySQL as a component of an application can be very powerful.
Perl Database Technologies

• A number of third-party CPAN modules provide access to MySQL from within a Perl program.
  – One such module is Net::MySQL by Hiroyuki Oyama, which provides a stable programming interface to MySQL functionality.
• In fact, nearly every database system provides a specific technology for programmers to use when programming their particular database.
  – This technology is referred to as an API, an application programming interface.
• Unfortunately, the effort expended in learning how to use Net::MySQL is of little use when a program has to be written to interface with Oracle or Sybase

Preparing Perl

DBI and DBD::mysql modules need to be installed
S man DBI
S man DBD::mysql
S find perl -T -e "@INC = @INC ". ".name = " pm "; print grep DBI.pm
S find perl -T -e "@INC = @INC ". ".name = " pm "; print grep mysql.pm
S locate DBI.pm
S locate mysql.pm

DBI (previously called DBperl) is a database independent interface module for Perl.
DBD: Data Base Description

Checking the DBI installation

#!/ /usr/bin/perl -w
# check_drivers - check which drivers are installed with DBI.
use strict;
use DBI qw(:utils);
use constant DATABASE => "DBI:mysql:MER";
use constant DB_USER => "bbp";
use constant DB_PASS => "passwordhere";
my $dbh = DBI->connect( DATABASE, DB_USER, DB_PASS )
  or die "Connect failed: ", $DBI::errstr,
my $sql = "show tables";
my $sth = $dbh->prepare( $sql );
$sth->execute;
print dump_results( $sth ), 
$sth->finish;
$dbh->disconnect;

Programming Databases With DBI

#!/ /usr/bin/perl -w
# show_tables - list the tables within the MER database.
# Uses "DBI::dump_results" to display results.
use strict;
use DBI qw( :utils );
use constant DATABASE => "DBI:mysql:MER";
use constant DB_USER => "bbp";
use constant DB_PASS => "passwordhere";
my $dbh = DBI->connect( DATABASE, DB_USER, DB_PASS )
  or die "Connect failed: ", $DBI::errstr,
my $sql = "show tables";
my $sth = $dbh->prepare( $sql );
$sth->execute;
print dump_results( $sth ), 
$sth->finish;
$dbh->disconnect;

Perl Database Technologies

• The DBI module provides a database independent interface for Perl.
  – By providing a generalised API, programmers can program at a “higher level” than the API provided by the database system, in effect insulating programs from changes to the database system.
• To connect the high-level DBI technology to a particular database system, a special driver converts the general DBI API into the database system-specific API.
  – These drivers are implemented as CPAN modules.

• DATABASE
  – Identifies the data source to use
• DB_USER
  – Identifies the username to use when connecting to the data source
• DB_PASS
  – Identifies the password to use when authenticating to the data source
Be sure to adhere to any established naming conventions within a programming community.

**The Sequence Retrieval System**

- Sequence Retrieval System (SRS) is a web-based database integration system that allows for the querying of data contained in a multitude of databases, all through a single user interface.
- This makes the individual databases appear as if they are really one big relational database, organised with different subsections:
  - one called SWISS-PROT,
  - one called EMBL,
  - one called PDB,
  - ...

**The Sequence Retrieval System**

- SRS makes it very easy to query the entire data set, using common search terms that work across all the different databases, regardless of what they are.
- Everything contained within the SRS is “tied together” by the web-based interface.
- Figure in the next slide is the database selection page from the EBI’s SRS web-site, which can be navigated to from the following Internet address:
  - [http://srs.ebi.ac.uk](http://srs.ebi.ac.uk)
  - SRS is a trademark and the intellectual property of Lion Bioscience

**EBI's SRS Database Selection Page**

**The Sequence Retrieval System**

- SRS is important for two reasons:
  - It is a useful and convenient service that every Bioinformatician should know about.
  - It is an excellent example of what can be created when the World Wide Web, databases and programming languages are combined.

- Warning:
  - Don’t create a new data format unless absolutely necessary.
  - Use an existing format whenever possible
Web Technologies

- The WWW was invented in 1991 by Tim Berners-Lee.
- The ability to publish data and applications on the Internet, in the form of custom web pages, is now considered an essential skill in many disciplines, including Biology.
- The development infrastructure of the World Wide Web (WWW) is well established and well understood.
- There is a standard set of infrastructural components (as suggested by Tim Berners-Lee):
  - The letters that form the acronym are taken from the words HyperText Mark-up Language, or HTML.
  - The transport protocol employed by the WWW is called Hypertext Transfer Protocol (HTTP).
  - Other components turn the standard web development infrastructure into a dynamic and powerful application development environment.
  - One of the reasons the WWW is so popular is the fact that creating content is so straightforward.
  - Adding a programming language into the mix allows even more to be accomplished.

The Web Development Infrastructure

- The web server
  - a program that when loaded onto a computer system provides for the publication of data and applications
  - often referred to collectively as content
  - Examples (apache, Jigsaw, and Microsoft’s IIS)
- The web client
  - a program that can request content from a web server and display the content within a graphical window, providing a mechanism whereby user can interact with the contents
  - The common name for the web client is web browser
  - Examples (Chrome, Mozilla, MS Internet Explorer, KDE Konqueror, Opera, Lynx, …)

Additional components

- Client-side programming
  - a technology used to program the web client, providing a way to enhance the user’s interactive experience.
  - Java applets, JavaScript, Macromedia Flash, …
- Server-side programming
  - a technology used to program the web server, providing a mechanism to extend the services provided by the web server.
  - Java Servlets, JSP, Python, ASP, PHP, Perl, …
- Backend database technology
  - a place to store the data to be published, which is accessed by the server-side programming technology.
  - MySQL, …

The Web Development Infrastructure

- Transport protocol
  - the “language” that the web server and web client use when communicating with each other.
  - Think of this as the set of rules and regulations to which the client and server must adhere.
  - The transport protocol employed by the WWW is called Hypertext Transfer Protocol (HTTP).
- The content
  - the data and applications published by the web server
  - this is textual data formatted to conform to one of the HyperText Mark-up Language standards (HTML)
  - HTML can be enhanced with embedded graphics.
  - Data published in the form of HTML is often referred to as HTML pages or web pages.

Additional components

- The acronym LAMP is used to describe the favoured WWW development infrastructure of many programmers.
  - The letters that form the acronym are taken from the words Linux, Apache, MySQL and Perl/Python/PHP.
- The additional components turn the standard web development infrastructure into a dynamic and powerful application development environment.
- One of the reasons the WWW is so popular is the fact that creating content is so straightforward.
- Adding a programming language into the mix allows even more to be accomplished.

Creating Content For The WWW

- There are a number of techniques employed to create HTML:
  - Creating content manually
    - Any text editor can be used to create HTML, since HTML is mostly text.
    - Special tags within the text guide the web browser when it comes to displaying the web page on screen.
    - The tags are also textual and any text editor can produce them
    - Advantages and disadvantages:
      - Provides the maximum amount of flexibility as the creator has complete control over the process.
      - It can also be advantageous to know what’s going on behind the scenes, so learning HTML is highly recommended.
      - Can be time-consuming and tedious, as the creator of the page has to write the content as well as decide which tags to use and where.
Creating Content For The WWW

- Creating content visually
  - Special-purpose editors can create HTML pages visually, displaying the web page as it will appear in the web browser as it is edited.
    - Netscape Composer, Microsoft FrontPage and Macromedia Dreamweaver, ...
  - Advantages and disadvantages:
    - no need to know anything about HTML.
    - The editor adds the required tags to the text that’s entered by the user.
    - unnecessary tags added.
    - HTML pages are larger.

Creating Content For The WWW

- Creating content dynamically
  - Since HTML is text, it is also possible to create HTML from a program.
  - Advantages and disadvantages:
    - HTML pages produced in this way can sometimes be useful when combined with a web server that allows for server-side programming of a backend database.
    - needs a web page creator to write a program to produce even the simplest of pages.

A Simple HTML Page

- Content of simple-m.html created manually

```html
<HTML>
<HEAD>
<TITLE>A Simple HTML Page</TITLE>
</HEAD>
<BODY>
This is as simple a web page as there is.
</BODY>
</HTML>
```

A Simple HTML Page

- Content of simple-k.html created visually
  - by using KompoZer
    - http://www.kompozer.net/

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
<title>A Simple HTML Page</title>
</head>
<body>
This is as simple a web page as there is.
</body>
</html>
```

Producing HTML

- Producing HTML with a Perl program using a HERE document:

```perl
#!/usr/bin/perl -w

# produce_simple produces the "simple.html" web page using a HERE document.
use strict;
print <<WEBPAGE;
<HTML>
<HEAD>
<TITLE>A Simple HTML Page</TITLE>
</HEAD>
<BODY>
This is as simple a web page as there is.
</BODY>
</HTML>
WEBPAGE
```

Producing HTML

- HTML file produced by the program:

```html
<HTML>
<HEAD>
<TITLE>A Simple HTML Page</TITLE>
</HEAD>
<BODY>
This is as simple a web page as there is.
</BODY>
</HTML>
```
Producing HTML, cont.

- Another version of HTML generation
  - written to use Perl’s standard CGI module

```perl
#!/usr/bin/perl -w

# produce_simpleCGI - produces the "simple.html" web page using
# Perl's standard CGI module.
use strict;
use CGI qw( :standard );
print start_html( 'A Simple HTML Page' ),
  "This is as simple a web page as there is.",
end_html;
```

- The CGI module is designed to make the production of HTML as convenient as possible.
- `start_html` subroutine produces the tags that appear at the start of the web page.
- `end_html` subroutine produces the following HTML, representing tags that conclude a web page:
  ```html
  </body></html>
  ```

Results from produce_simpleCGI

- HTML file produced by the program:

```html
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"
xml:lang="en-US">
<head>
<title>A Simple HTML Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
</head>
<body>
This is as simple a web page as there is.
</body>
</html>
```

The dynamic creation of WWW content

- When the web page includes content that is not static, it is referred to as dynamic web page.
  - For example a page including current date and time
- It is not possible to create a web page either manually or visually that includes dynamic content, and
  - this is where server side programming technologies come into their own.

Static creation of WWW content

- `simple.html` web page is static
- If the web page is put on a web server it always appear in exactly the same way every time it is accessed.
  - It is static, and remains unchanged until someone takes the time to change it.
- It rarely makes sense to create such a web page with a program unless you have a special requirement.
  - Create static web pages either manually or visually

The dynamic creation of WWW content

```perl
#!/usr/bin/perl -wT

# whattimeisit - create a dynamic web page that includes the
# current date/time.
use strict;
use CGI qw( :standard );
print start_html( 'What Date and Time Is It?' ),
  "The current date/time is: ", scalar localtime,
end_html;
```

Extra staff at the start is optional. Extra tags tell the web browser exactly which version of HTML the web page conforms to. The CGI module includes these tags for web browser to optimise its behaviour to the version of HTML identified.
Results from whattimeisit ...

>perl -wT whattime.pl

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>What Date and Time Is It?</title>
</head>
<body>
The current date/time is: Thu Mar 29 18:56:17 2007</body>
</html>

 And some time later ...

>perl -wT whattime.pl

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml" lang="en-US">
<head>
<title>What Date and Time Is It?</title>
</head>
<body>
The current date/time is: Thu Mar 29 18:59:59 2007</body>
</html>

• This web page, if served up by a web server, changes with each serving, as it is dynamic.

• Note that use of the "T" command-line option at the start of the program.
  – This switches on Perl’s taint mode,
    • which enables a set of special security checks on the behaviour of the program.
  • If a server-side program does something that could potentially be exploited and, as a consequence, pose a security treat, Perl refuses to execute the program when taint mode is enabled.
  • Always enable "taint mode" for server-side programs
  • Test your web-site on localhost prior to deployment on the Internet

Sending Data To A Web Server

• Switch on taint mode on the Perl command line
• Use CGI module, importing (at least) the standard set of subroutines
• Ensure the first print statement within the program is "print header";
• Envelope any output sent to STDOUT with calls to the start_html and end_html subroutines
• Create a static web page to invoke the server-side program, providing input as necessary

Sending Data To A Web Server

#!/usr/bin/perl -wT
#
# The 'match_emblCGI' program - check a sequence against the EMBL
# database entry stored in the embl.data.out data-file on the web server.
#
use strict;
use CGI::perl::standard;

print header;
open EMBLENTRY, "embl.data.out" or die "No data-file: have you executed prepare_embl?

my $sequence = <EMBLENTRY>
close EMBLENTRY;

match_emblCGI, cont.

print start_html( "The results of your search are in!" );
p
if ( $sequence =~ /$to_check/ ) {
  print "Found. The EMBL data extract contains: <b>$to_check</b>.
}
else {
  print "Sorry. No match found for: <b>$to_check</b>."
}
end_html;
A Search HTML Page

```html
<HTML>
<HEAD>
<TITLE>Search the Sequence for a Match</TITLE>
</HEAD>
<BODY>
Please enter a sequence to match against:<p>
<FORM ACTION="/cgi-bin/match_emblCGI">
<p>
<textarea name="shortsequence" rows="4" cols="60"></textarea>
</p>
<p>
<input type="reset" value="Clear">
<input type="submit" value="Try it!">
</p>
</FORM>
</BODY>
</HTML>
```

Installing CGIs on a Web Server

```
su
$ cp mersearch.html /var/www/html
$ cp match_emblCGI /var/www/cgi-bin
$ chmod +x /var/www/cgi-bin/match_embl
$ cp embl.data.out /var/www/cgi-bin
$ <Ctrl-D>
```

The ``Search the Sequence for a Match'' web page

![Image](figMERSEARCH.png)

The ``Results of your search are in!'' web page

![Image](figMERSEARCHFOUND.png)

The ``Sorry! Not Found'' web page

![Image](figMERSEARCHSORRY.png)

Using a HERE document

```perl
print <<MERFORM;
Please enter another sequence to match against:<p>
<FORM ACTION="/cgi-bin/match_emblCGIbetter">
<p>
<textarea name="shortsequence" rows="4" cols="60"></textarea>
</p>
<p>
<input type="reset" value="Clear">
<input type="submit" value="Try it!">
</p>
</FORM>
MERFORM
```
Web Databases

Better version: "Results of your search are in!" web page

Searching all the entries in the dnas table

The "results" of the multiple search on the dnas table

Installing DB Multi-Search

```
$ su
$ cp mersearchmulti.html /var/www/html
$ cp db_match_emblCGI /var/www/cgi-bin
$ chmod +x /var/www/cgi-bin/db_match_emblCGI
$ cp /home/barryp/DbUtilsMER.pm /var/www/cgi-bin
$ <Ctrl-D>
```

Web Automation

Using Perl to automate web surfing
Why Automate Surfing?

• Imagine you have 100 sequences to check.
• If it takes average 1 minute to enter the sequence into text area, entering 100 sequences requires 100 minutes
• Why not automate it to save time

Perl module WWW::Mechanize allows programmer to automate interactions with any web-site

Strategy to follow when automating interactions with any web page

• Load the web page of interest into a graphical browser
• View the HTML used to display the web page by selecting the Page Source option from browser’s View menu
• Read the HTML and make a note of the names of the interface elements and form buttons that are of interest
• Write a Perl program that user WWW::Mechanize to interact with the web page (based on automatch, if needed)
• Use an appropriate regular expression to extract the interesting bits from the results returned from the web server

The automatch program

```perl
#!/usr/bin/perl -w

# The 'automatch' program - check a collection of sequences against 
# the 'mersearchmulti.html' web page.
use strict;
use constant URL => "http://pblinux.itcarlow.ie/mersearchmulti.html";
use WWW::Mechanize;
my $browser = WWW::Mechanize->new;
while ( my $seq = <> )
{
    chomp( $seq );
    print "Now processing: '$seq'.
    $browser->get( URL );
    $browser->form( 1 );
    $browser->field( "shortsequence", $seq );
    $browser->submit;
    if ( $browser->success )
    {
        my $content = $browser->content;
        while ( $content =~ m<tr align="CENTER"/>td>(.+?)</td><td>yes</td>\]</g )
        {
            print "Accession code: $1 matched '$seq'.
        }
    }
    else
    {
        print "Something went wrong: HTTP status code: ",
        $browser->status, "\n";
    }
}
```

Running the automatch program

```bash
$ chmod +x automatch
$ ./automatch sequences.txt
```

Results from automatch

Now processing: 'atccgaaggcggca'.
Now processing: 'aattc'.
Accession code: AF213017 matched 'aattc'.
Accession code: J01730 matched 'aattc'.
Accession code: M24940 matched 'aattc'.
Now processing: 'aatgggc'.
Now processing: 'aaattt'.
Accession code: AF213017 matched 'aaattt'.
Accession code: J01730 matched 'aaattt'.
Accession code: M24940 matched 'aaattt'.
Now processing: 'acgatccgcaagtagcaacc'.
Accession code: M15049 matched 'acgatccgcaagtagcaacc'.
Now processing: 'gggcccaaa'.
Now processing: 'atcgatcg'.
Now processing: 'tcatgcacctgatgaacgtgcaaaaccacag'.
Accession code: AF213017 matched 'tcatgcacctgatgaacgtgcaaaaccacag'.
Accession code: AF213017 matched 'ccaaat'.
Accession code: J01730 matched 'ccaaat'.
Accession code: M24940 matched 'ccaaat'.

Results from automatch ...

Accession code: AF213017 matched 'aatttc'.
Accession code: J01730 matched 'aatttc'.
Accession code: M24940 matched 'aatttc'.
Now processing: 'atccgaaggcggca'.
Accession code: AF213017 matched 'atccgaaggcggca'.
Now processing: 'aattc'.
Accession code: M15049 matched 'aattc'.
Accession code: AF213017 matched 'aattc'.
Now processing: 'aatgggc'.
Now processing: 'aaattt'.
Accession code: AF213017 matched 'aaattt'.
Accession code: AF213017 matched 'aaattt'.
Accession code: M24940 matched 'aaattt'.
Accession code: M24940 matched 'aaattt'.
Automate repetitive WWW interactions whenever possible