CMSC423: Bioinformatic Algorithms, Databases and Tools

In the news: Encode

DNA project interprets 'book of life'

By **Elizabeth Landau**, CNN
updated 2:41 PM EDT, Wed September 5, 2012

Scientists are learning more about how DNA works in our bodies.

STORY HIGHLIGHTS

- The ENCODE project finds more than 80% of genome has biochemical function
- There are 4 million sites in the genome where events occur, representing "switches"
- "This century, we are going to be working out how we make humans," a scientist says

(CNN) -- Our genes play a major role in making us who we are, but a lot of information about their function has been mysterious.

That's why an international team of researchers set out to figure out what the working parts of the human genome are, and what they mean for the human body as we know it.

The project is called the Encyclopedia of DNA Elements (ENCODE). When the Human Genome Project sequenced the human genome in

ARTICLE

doi:10.1038/nature11247

An integrated encyclopedia of DNA elements in the human genome

The ENCODE Project Consortium*

The human genome encodes the blueprint of life, but the function of the vast majority of its nearly three billion bases is unknown. The Encyclopedia of DNA Elements (ENCODE) project has systematically mapped regions of transcription, transcription factor association, chromatin structure and histone modification. These data enabled us to assign biochemical functions for 80% of the genome, in particular outside of the well-studied protein-coding regions. Many discovered candidate regulatory elements are physically associated with one another and with expressed genes, providing new insights into the mechanisms of gene regulation. The newly identified elements also show a statistical correspondence to sequence variants linked to human disease, and can thereby guide interpretation of this variation. Overall, the project provides new insights into the organization and regulation of our genes and genome, and is an expansive resource of functional annotations for biomedical research.

CMSC423: Bioinformatic Algorithms, Databases and Tools

Project Specification and Part 1

CMSC423: Bioinformatic Algorithms, Databases and Tools

Writing bioinformatics software

Libraries & misc.

Libraries/utilities

- Bio::Perl (Perl)
- BioJava (Java)
- BioPython (Python)
- BioRuby (Ruby)
- seqAn (C++)
- Bioconductor (R)

Bio::Perl

- http://www.bioperl.org

```
use Bio::Perl;

my $seq = read_sequence(“mytest.fa”, “fasta”);
my $gbseq = read_sequence(“mytest.gb”, “genbank”);

write_sequence(“>test.fasta”, 'fasta', $gbseq);
```

' vs “ ?

Bio::Perl

- Find sequences longer than 500 letters

```
use Bio:Perl;


while ($seq = read_sequence(“test.fa”, 'fasta')) {
    if ($seq ->length() > 500) {
        print $seq->primary_id(), “\n”;
    }
}
```

Bio::Perl

- Other useful stuff

```
$seqio = new Bio::SeqIO(-format => 'largefasta',
                        -file   => 't/data/genomic-seq.fasta');
$pseq = $seqio->next_seq();

$gb = new Bio::DB::GenBank;
$seq1 = $gb->get_Seq_by_id('MUSIGHBA1');
```

etc...

BioPython

- http://www.biopython.org

```
from Bio import SeqIO
handle = open(“file.fasta”)
seq_record = SeqIO.parse(handle, “fasta”)

SeqIO.write(my_records, handle2, "fasta")
```

BioPython

```
from Bio import SeqIO
handle = open("test.fasta")
for seq_record in SeqIO.parse(handle, "fasta") :
    if len(seq_record) > 500 :
        print seq_record.id
handle.close()
```

BioPython...more

- Same as Bio::Perl:
  - can directly connect to databases
  - various sequence manipulations (reverse complement, translate, etc.)
  - basic bioinformatics algorithms
  - etc.

BioJava

- http://www.biojava.org

```
import org.biojava.bio.*;
String filename = args[0];
    BufferedInputStream is =
        new BufferedInputStream(new FileInputStream(filename));
    //get the appropriate Alphabet
    Alphabet alpha =
  AlphabetManager.alphabetForName(args[1]);

    //get a SequenceDB of all sequences in the file
    SequenceDB db = SeqIOTools.readFasta(is, alpha);
```

BioJava

```
BufferedReader br =
        new BufferedReader(new FileReader(args[0]));

 String format = args[1];
 String alphabet = args[2];

 SequenceIterator iter =
 (SequenceIterator)SeqIOTools.fileToBiojava(format,alphabet, br);
while (iter.hasNext()){
    Sequence seq = iter.nextSequence();
    if (seq.length() > 500) {System.out.println(seq.getName());}
}
```

BioJava...more

- Same as Bio::Perl:
  - can directly connect to databases
  - various sequence manipulations (reverse complement, translate, etc.)
  - basic bioinformatics algorithms
  - etc.

BioRuby

- http://www.bioruby.org

```
require 'bio'

input_seq = ARGF.read      # reads all files inarguments

my_naseq = Bio::Sequence::NA.new(input_seq)
```

BioRuby

```
#!/usr/bin/env ruby

require 'bio'

ff = Bio::FlatFile.new(Bio::FastaFormat, ARGF)
ff.each_entry do |f|
  if f.length > 500
    puts f.entry_id
  end
end
```

BioRuby...more

- Same as Bio::Perl:
  - can directly connect to databases
  - various sequence manipulations (reverse complement, translate, etc.)
  - basic bioinformatics algorithms
  - etc.

SeqAn

- http://www.seqan.de

```
#include <seqan/sequence.h>
#include <seqan/file.h>

using namespace seqan;
using namespace std;

String <Dna> seq;
String<char> name;
fstream f;
f.open(“test.fasta”);
readMeta(f, name, Fasta());
readMeta(f, seq, Fasta());
```

SeqAn

```
String <Dna> seq;
String<char> name;
fstream f;
f.open(“test.fasta”);
while (! f.eof()){
    readMeta(f, name, Fasta());
    readMeta(f, seq, Fasta());
    if (length(seq)){
        cout << name << endl;
    }
}
```

SeqAn...more

- Not quite as much as Perl/Java/Python, but still lots of utilities (including graph algorithms)

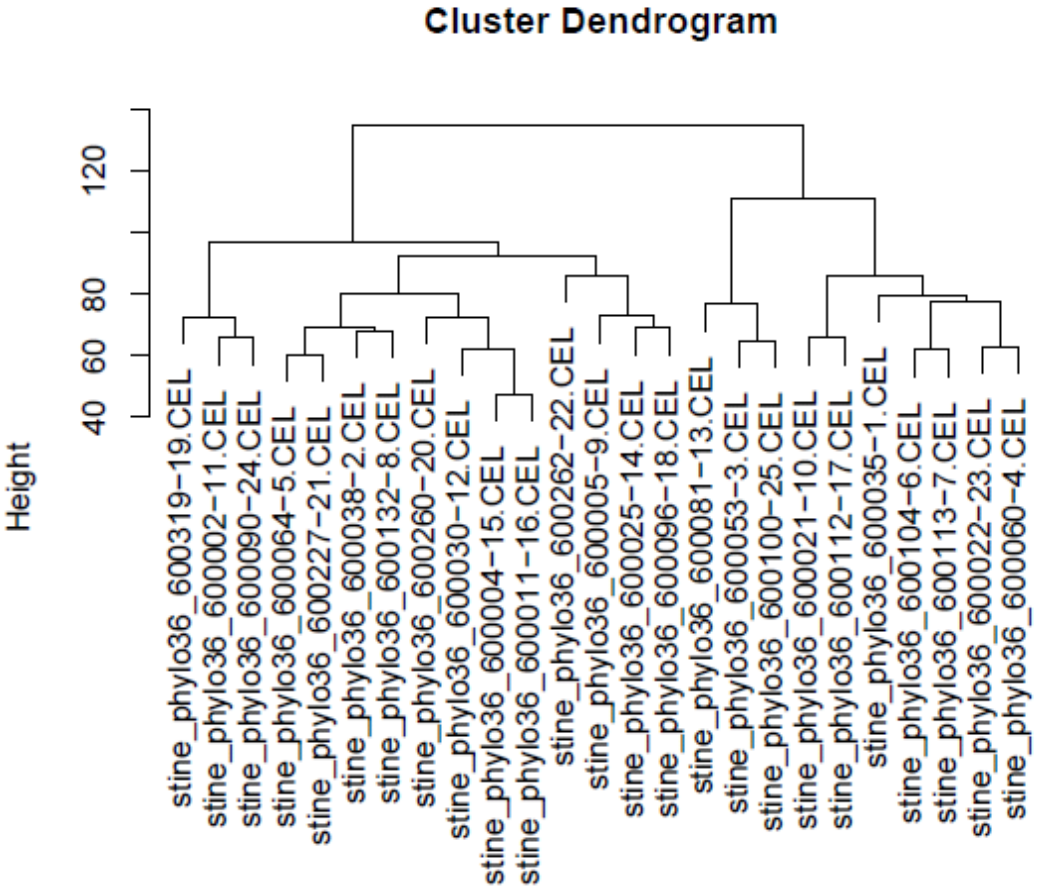
R/BioConductor

- http://www.bioconductor.org
- Mainly for statistical applications, e.g. microarray analysis

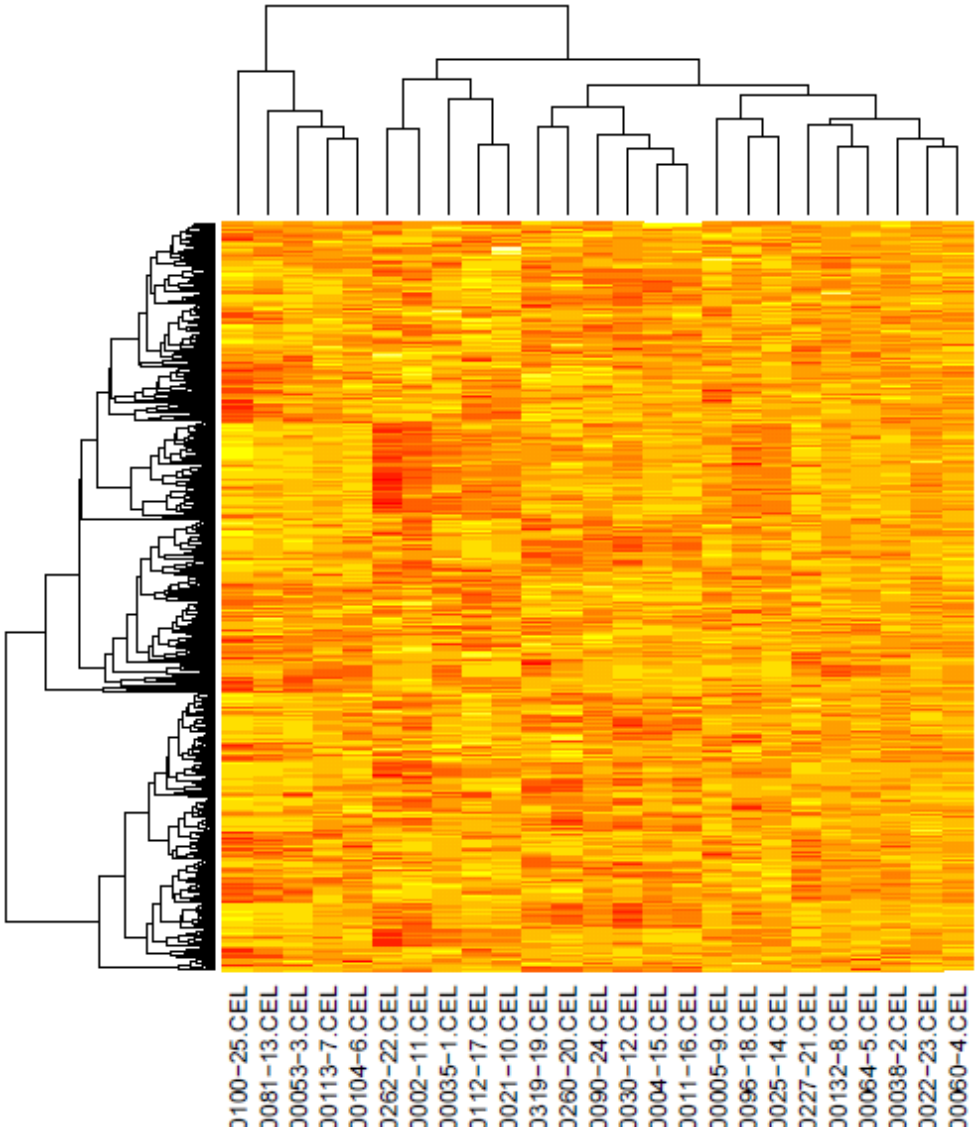
```
library("affy")
library("geneplotter")
library("gplots")


data <- ReadAffy()
eset <- rma(data)
e <- exprs(eset)
heatmap.2(e, margin=c(15,15), trace="none",
   col=redgreen(25), cexRow=0.5)
```

R/BioConductor

- Book has lots of examples
- Worth learning more about it – easy to do various cool things



R... more cool stuff



Programming for bioinformatics

- Details of specialized libraries beyond scope of course
- Good software engineering practices are essential
- Often, “correct” is undefined – output of program defines correctness
- Pitfalls – e.g. papers retracted from Science due to software bugs

- Key – be proactive and learn by yourselves from online resources!

http://www.biomedcentral.com/content/pdf/1471-2105-9-82.pdf

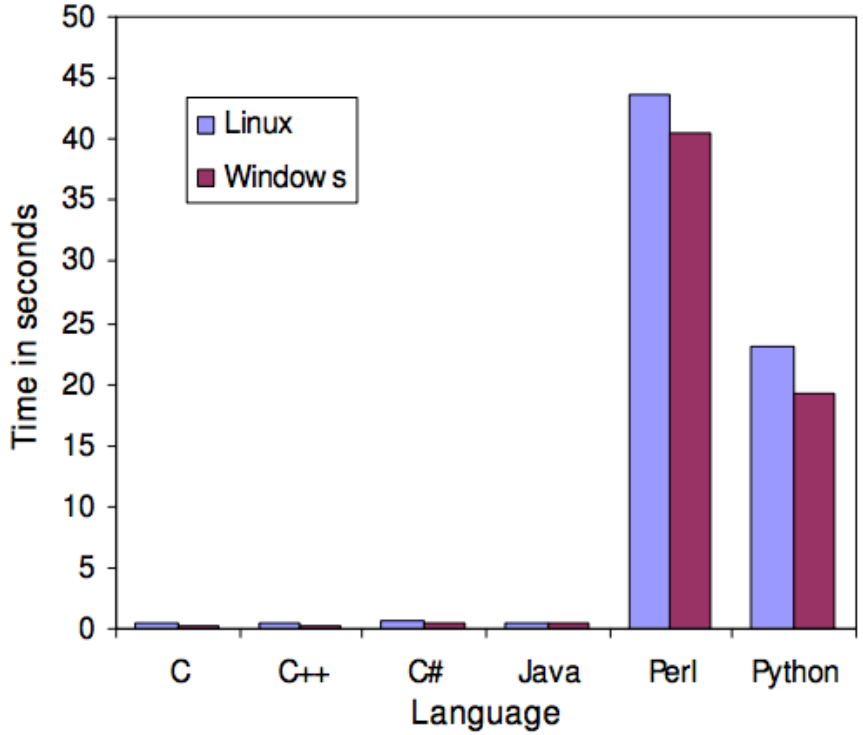


Figure 1
Speed comparison of the global alignment program. Speed comparison of the global alignment algorithm using a gap penalty of 10 implemented in C, C++, C#, Java, Perl and Python. The programs were run on Linux and Windows platforms. Two DNA sequences of 3216 bp and 3217 bp were used.

http://www.biomedcentral.com/content/pdf/1471-2105-9-82.pdf

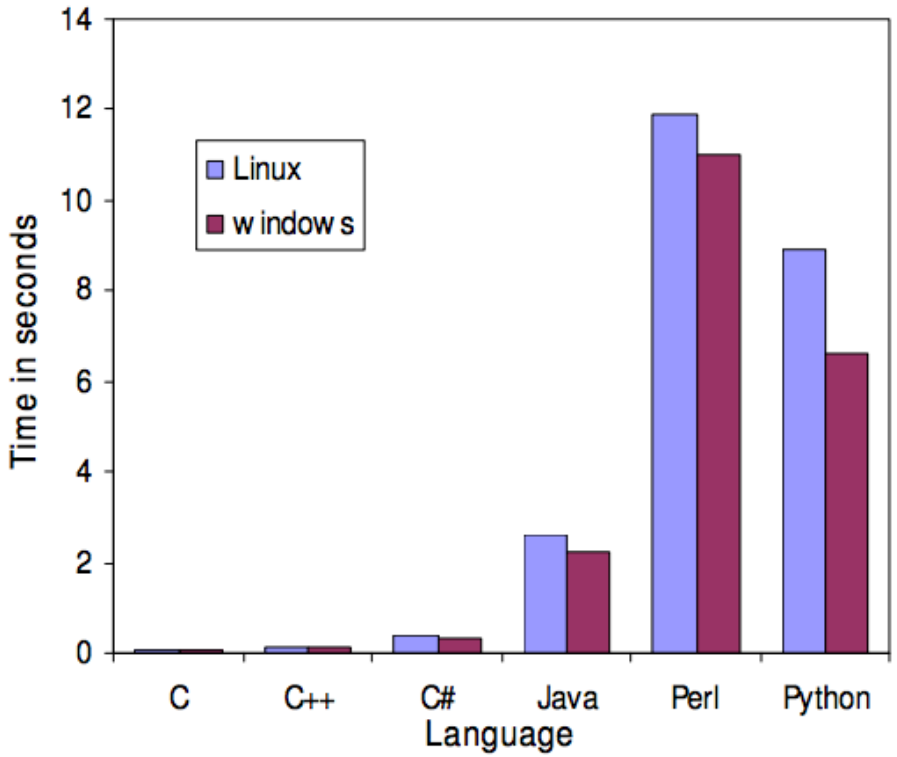


Figure 2
Speed comparison of the Neighbor-Joining program. Speed comparison of the Neighbor-Joining algorithm using the Jukes-Cantor evolutionary model implemented in C, C++, C#, Java, Perl and Python. The programs were run on Linux and Windows platforms. The input file was an alignment of 76 DNA sequences.

http://www.biomedcentral.com/content/pdf/1471-2105-9-82.pdf

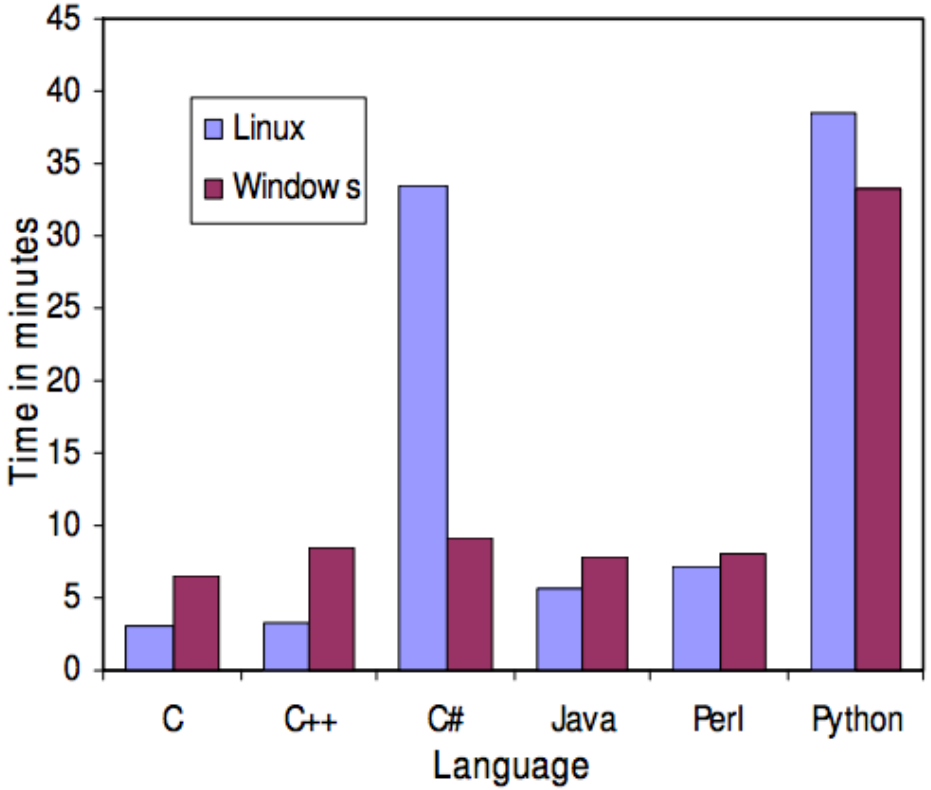


Figure 3
Speed comparison of the BLAST parsing program. Speed comparison of the BLAST parsing program implemented in C, C++, C#, Java, Perl and Python. The programs were run on Linux and Windows platforms. The input file was a 9.8 Gb file from a BLASTP run.

http://www.biomedcentral.com/content/pdf/1471-2105-9-82.pdf

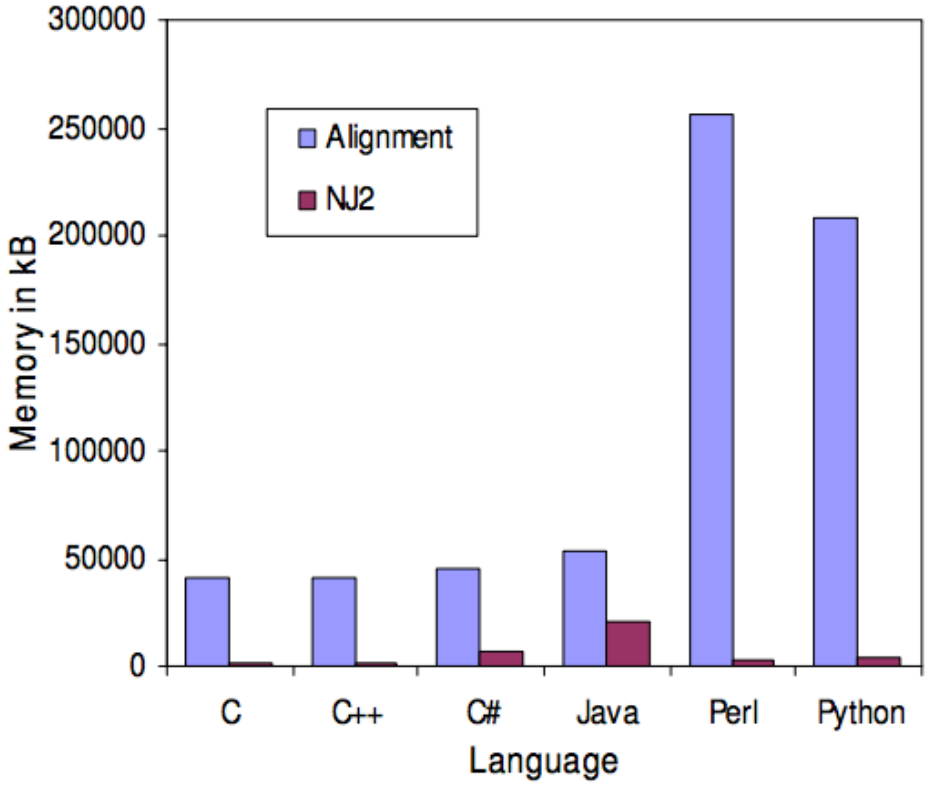


Figure 4
Memory usage comparison of the Neighbor-Joining and global alignment programs. Memory usage comparison for the Neighbor-Joining and global alignment programs implemented in C, C++, C#, Java, Perl and Python. The programs were run on a Linux platform.

http://www.biomedcentral.com/content/pdf/1471-2105-9-82.pdf

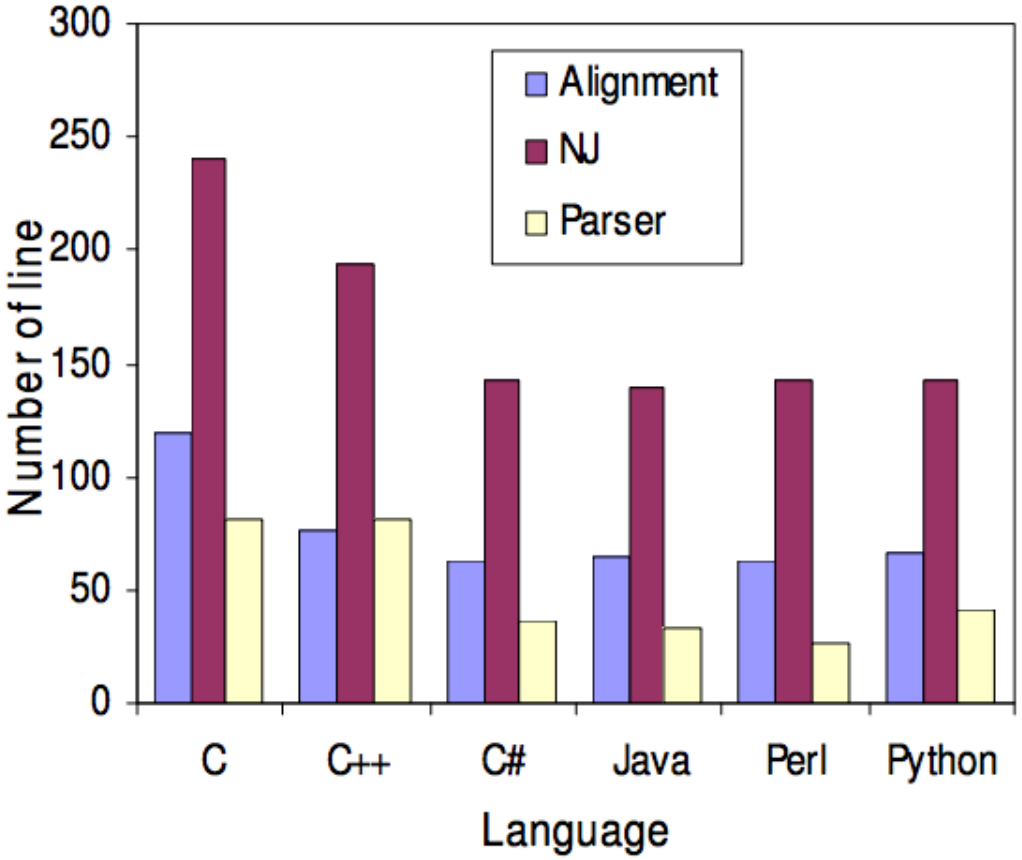


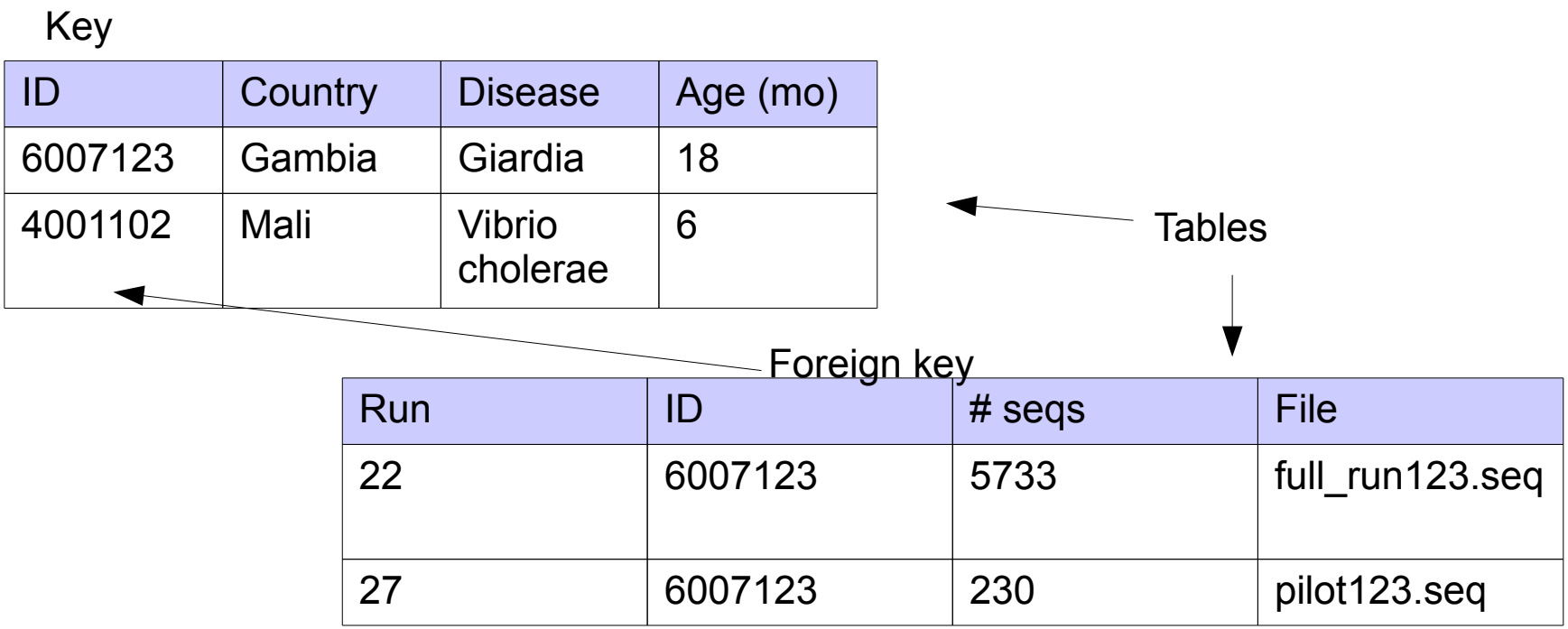
Figure 5
Number of lines for each program. Number of lines for the global alignment, BLAST parser and Neighbor-Joining programs implemented in C, C++, C#, Java, Perl and Python.

CMSC423: Bioinformatic Algorithms, Databases and Tools

Biological databases

What's a database?

- Take CMSC424 for in-depth view
- Essentially a collection of Excel sheets or tables (note: only true for the “relational model” - most popular)



Biological databases

- General
  - GenBank - US
  - EMBL - Europe
- Specialized by data type
  - NCBI SRA – raw sequencing data
  - SwissProt – curated protein information
  - KEGG – biological pathways
  - Gene Expression Omnibus – microarray data
- Specialized by organism
  - ZFIN – zebrafish
  - SGD – yeast
  - WormBase - worms

What data gets stored?

- DNA
  - string of letters
  - quality information, maybe chromatograms
  - location of genes (ranges along a chromosome)
- Proteins
  - string of letters
  - protein domains
  - 3D coordinates of each atom
- Pathways
  - graph of interactions between genes

For all – often store link to scientific articles related to data

How the data get accessed

- Gene by gene/object by object – targeted at manual inspection of data
  - usually lots of clicking involved
  - simple search capability
  - similarity searches in addition to text queries
- Bulk – targeted at computational analyses
  - often programmatic access through web server
  - most frequently – just bulk download (ftp)

NCBI - National Center for Biotech. Info.

- Virtually all biological data generated in the US gets stored here!
- One-stop-shop for biological data
- Primarily focused on gene-by-gene analyses
- Provides simple scripts for programmatic access
- Provides ftp access for bulk downloads

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

EMBL European Molecular Biology Lab.

- European version of NCBI
- BioMart query builder

http://www.ebi.ac.uk/embl/

Expasy proteomics server

- Home of Swisprot and other useful information on proteins

http://www.expasy.org

Programmatic database access

```
use DBI;

my $dbh = DBI->connect("dbi:Sybase:server=SERV;packetSize=8092",
                       "anonymous", "anonymous");
if (! defined $dbh) {
        die ("Cannot connect to server\n");
}

my $mysqlqry = <STDIN>;

$dbh->do("set textsize 65535");

my $qh = $dbh->prepare($mysqlqry) || die ("Cannot prepare\n");
$qh->execute() || die ("Cannot execute\n");

while (my @row = $qh->fetchrow()){
        processrow($row);
}
```

BioPython and GenBank

```
from Bio import SeqIO
gb_file = "NC_005213.gbk"
gb_record = SeqIO.read(open(gb_file,"r"), "genbank")
print "Name %s, %i features" % (gb_record.name, len(gb_record.features))
print repr(gb_record.seq)
```

Kyoto Encyclopedia of Genes & Genomes

- Central repository of pathway information

http://www.genome.jp/kegg/

Genome browsers

- UCSC Genome Browser – http://genome.ucsc.edu
- ENSEMBL Genome Browser – http://www.ensemble.org
- Gbrowse http://www.gmod.org

NCBI programmatic access

- http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html
  - must write your own HTTP client (LWP Perl module helps)
  - queries go directly to web server
  - data returned in XML
- http://www.ncbi.nlm.nih.gov/Traces/trace.cgi?cmd=show&f=doc&m=obtain&s=stips
  - stub script provided (query_tracedb)
  - queries still go through web server
  - data returned in a variety of user selected formats
- For both, limits are set on the amount of data retrieved, e.g. less than 40,000 records at a time
- Download procedure:
  - figure out # of records to be retrieved ("count" query)
  - read data in allowable chunks
  - combine the chunks

Biological Ontologies

- Gene Ontology. http://www.geneontology.org
  The Gene Ontology project provides a controlled vocabulary to describe gene and gene product attributes in any organism. (text from GO homepage)

- Note: similar to semantic web
- GO not the only one: http://www.obofoundry.org

Exercises

- Create a FASTA file containing all recA genes found in bacteria. Note: you can use a combination of manual queries and additional scripts (sometimes an NCBI query doesn't quite return what you want)