Perl & Bioinformatics

Intros to...
Genetics
BioPerl
SeqLab.net

by Jay Hannah

Jay?

Jay Hannah
12 years of Perl, DBs, Internet. Bio newb.

Collaborative Laboratory for
Applied Bioinformatics (CLAB)
http://clab.ist.unomaha.edu/CLAB/

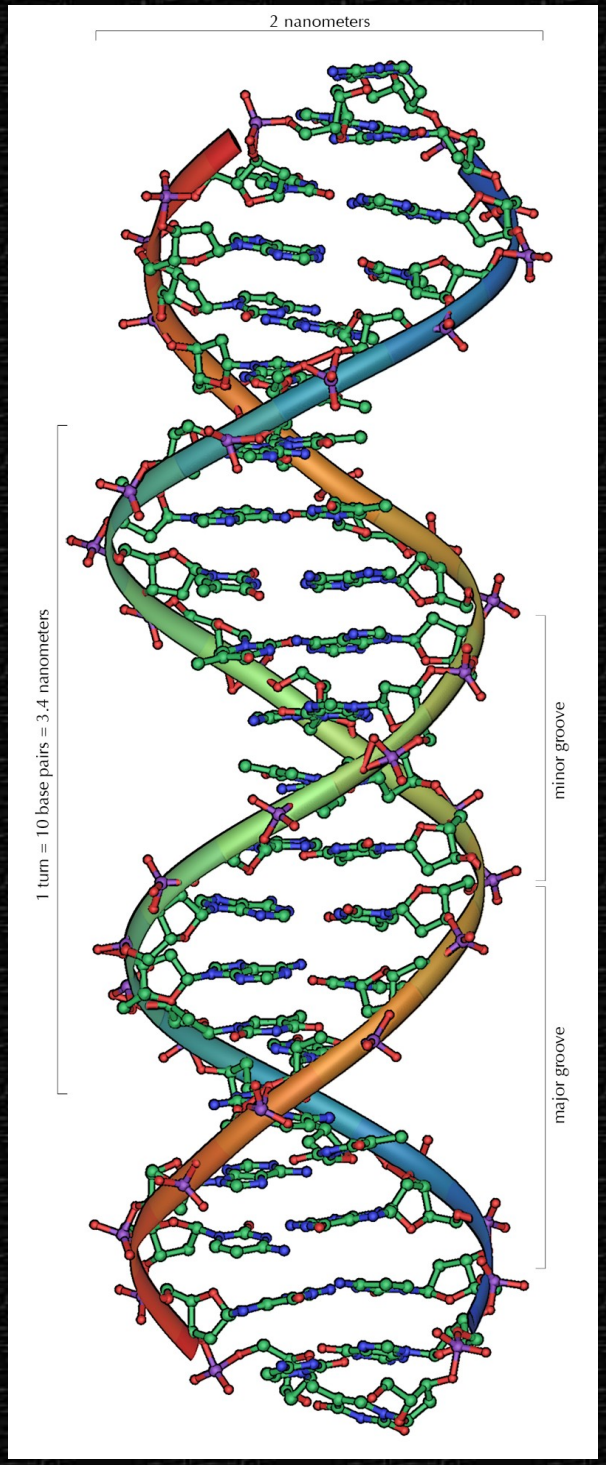
Join us!

DNA

http://en.wikipedia.org/wiki/DNA

This entire structure can be represented in only 21 characters (bases):

ACGTAGGATCGGATACGATAG



Collaborative Laboratory for Applied Bioinformatics (CLAB)
http://clab.ist.unomaha.edu/CLAB/ Join us!

by Jay Hannah

The Human Genome

- 3 billion DNA base pairs (A, C, G, or T)
  - Fully extended, the DNA from a single cell would have a total length of almost 6 feet.
  - All the DNA in your cells could reach the moon ...6000 times!
- 24 distinct chromosomes
- Estimated 20,000–25,000 genes

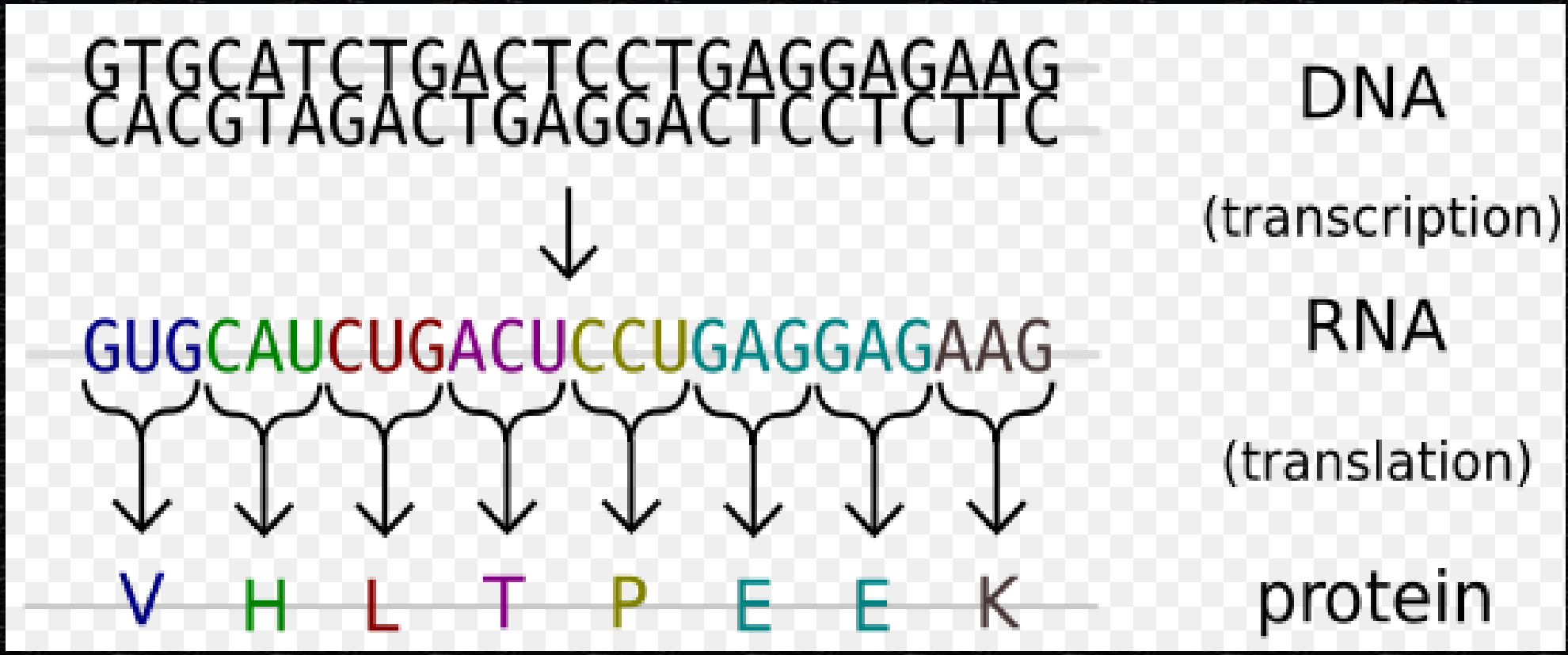
http://en.wikipedia.org/wiki/Human_genome
http://www.rothamsted.ac.uk/notebook/courses/guide/dnast.htm

The Human Genome

- Only 2.5% of DNA is different between humans and mice. Only 1% different from chimpanzee. [1]
- "We share half our genes [DNA] with the banana." [2]
- DNA is the blueprint of ALL life. You grew from a single cell to an adult human. What made you you? Why aren't you me? Or a chimp? Or a banana tree, a whale shark, plankton, a clover, or a giant redwood? Answer: proteins.

1. Mural, R.J., et al., Science, v. 296, May 31, 2002, p. 1661.
2. May, R., Quoted in Coglan & Boyce, New Scientist 167 (July 1):5, 2000

The “Central Dogma of Molecular Biology”



http://en.wikipedia.org/wiki/Image:Genetic_code.svg

NA / DNA

DNA is long strings of nucleotides.

There are 4 types of nucleotide (bases):

A - Adenine
C - Cytosine
G - Guanine
T - Thymine

C <=> G A <=> T

Transcription

DNA => RNA
A => U
T => A
G => C
C => G

In RNA, thymine (T) is replaced by uracil (U).

AA / protein

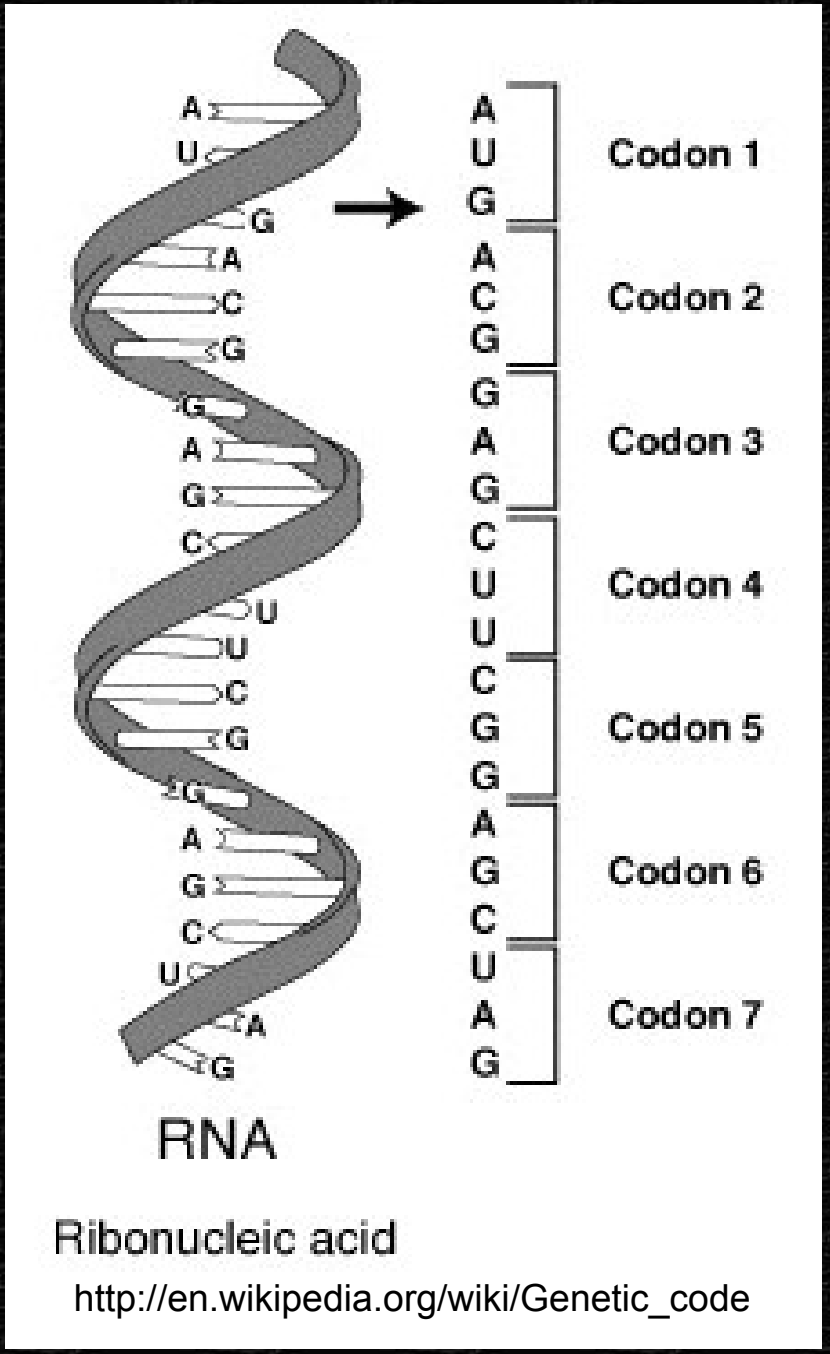
Proteins are long strings of amino acids.
There are 21 amino acids.

Alanine	A	Ala	Asparagine	N	Asn
Cysteine	C	Cys	Proline	P	Pro
Aspartic acid	D	Asp	Glutamine	Q	Gln
Glutamic acid	E	Glu	Arginine	R	Arg
Phenylalanine	F	Phe	Serine	S	Ser
Glycine	G	Gly	Threonine	T	Thr
Histidine	H	His	Selenocysteine	U	Sec
Isoleucine	I	Ile	Valine	V	Val
Lysine	K	Lys	Tryptophan	W	Trp
Leucine	L	Leu	Tyrosine	Y	Tyr
Methionine	M	Met			

Translation

RNA => protein: “translation”

(remember earlier we covered
DNA => RNA: “transcription”)

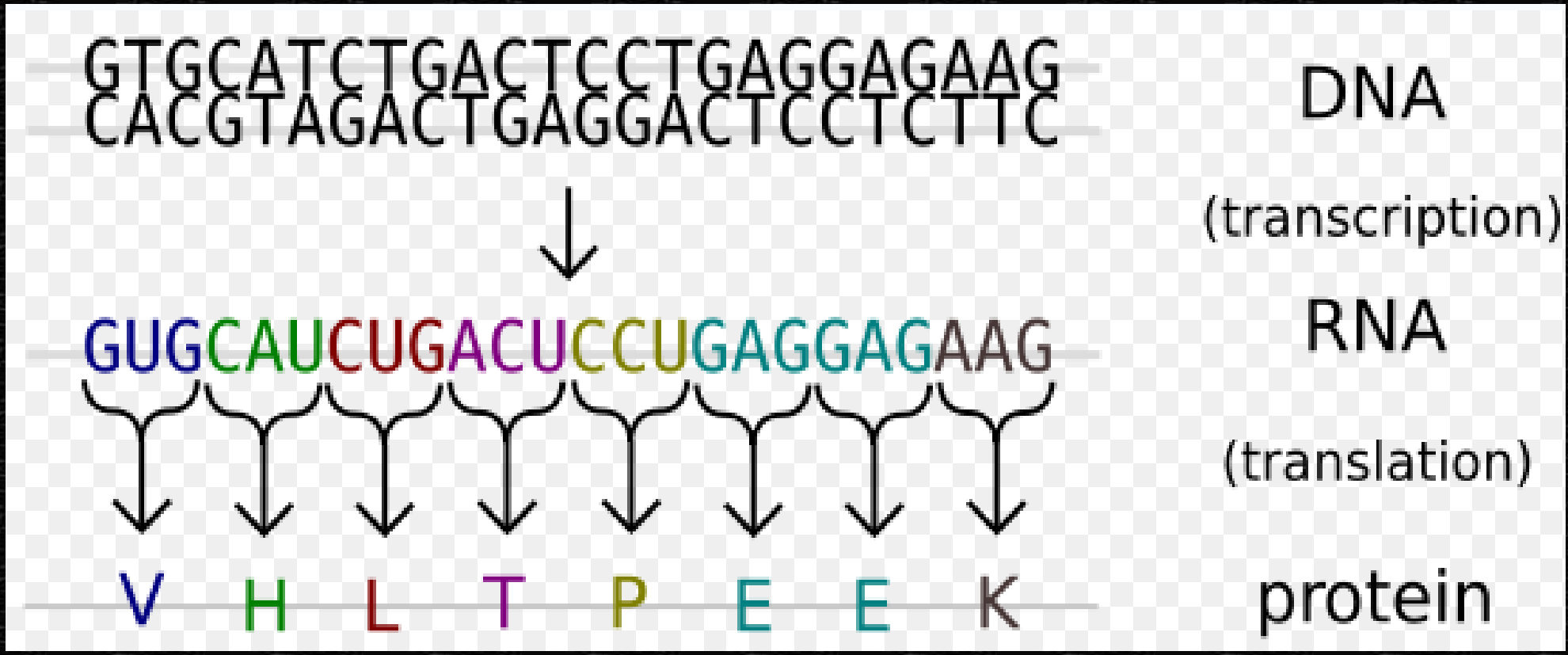


RNA codon table

This table shows the 64 codons and the amino acid each codon codes for. The direction is 5' to 3'.

		2nd base			
		U	**C**	**A**	**G**
1st base	**U**	UUU (Phe/F)Phenylalanine UUC (Phe/F)Phenylalanine UUA (Leu/L)Leucine UUG (Leu/L)Leucine	UCU (Ser/S)Serine UCC (Ser/S)Serine UCA (Ser/S)Serine UCG (Ser/S)Serine	UAU (Tyr/Y)Tyrosine UAC (Tyr/Y)Tyrosine UAA Ochre (*Stop*) UAG Amber (*Stop*)	UGU (Cys/C)Cysteine UGC (Cys/C)Cysteine UGA Opal (*Stop*) UGG (Trp/W)Tryptophan
	C	CUU (Leu/L)Leucine CUC (Leu/L)Leucine CUA (Leu/L)Leucine CUG (Leu/L)Leucine	CCU (Pro/P)Proline CCC (Pro/P)Proline CCA (Pro/P)Proline CCG (Pro/P)Proline	CAU (His/H)Histidine CAC (His/H)Histidine CAA (Gln/Q)Glutamine CAG (Gln/Q)Glutamine	CGU (Arg/R)Arginine CGC (Arg/R)Arginine CGA (Arg/R)Arginine CGG (Arg/R)Arginine
	A	AUU (Ile/I)Isoleucine AUC (Ile/I)Isoleucine AUA (Ile/I)Isoleucine AUG (Met/M)Methionine, *Start*[1]	ACU (Thr/T)Threonine ACC (Thr/T)Threonine ACA (Thr/T)Threonine ACG (Thr/T)Threonine	AAU (Asn/N)Asparagine AAC (Asn/N)Asparagine AAA (Lys/K)Lysine AAG (Lys/K)Lysine	AGU (Ser/S)Serine AGC (Ser/S)Serine AGA (Arg/R)Arginine AGG (Arg/R)Arginine
	G	GUU (Val/V)Valine GUC (Val/V)Valine GUA (Val/V)Valine GUG (Val/V)Valine	GCU (Ala/A)Alanine GCC (Ala/A)Alanine GCA (Ala/A)Alanine GCG (Ala/A)Alanine	GAU (Asp/D)Aspartic acid GAC (Asp/D)Aspartic acid GAA (Glu/E)Glutamic acid GAG (Glu/E)Glutamic acid	GGU (Gly/G)Glycine GGC (Gly/G)Glycine GGA (Gly/G)Glycine GGG (Gly/G)Glycine

The “Central Dogma of Molecular Biology”



http://en.wikipedia.org/wiki/Image:Genetic_code.svg

The “Central Dogma of Molecular Biology”

```
use Bio::Seq;       # BioPerl!
$seq = Bio::Seq->new(
    -seq => 'GTGCATCTGACTCCTGAGGAGAAG',
    -id  => 'JAY1',
);
print $seq->translate->seq . "\n";

# VHLTPEEK
```

BioPerl

- www.bioperl.org
- “A collection of Perl modules that facilitate the development of Perl scripts [programs] for bioinformatics applications.” - Wikipedia
- Humongous toolbox of problem solutions that take care of ugly details I don't want to try to understand yet. - Jay
- Use “bioperl-live” - Jay

BLAST

- Basic Local Alignment Search Tool
- Given a sequence, search one or millions of other sequences looking for similarities. (Google for white coats.)
  - Why? Similar proteins might do similar things.
  - Why are sick people sick? Grab samples and search databases looking for known bugs.
  - Compare sick people with healthy people. Where are the differences? What can we do about them?
  - Steal neat tricks from other critters.

Microarray

http://en.wikipedia.org/wiki/DNA_microarray

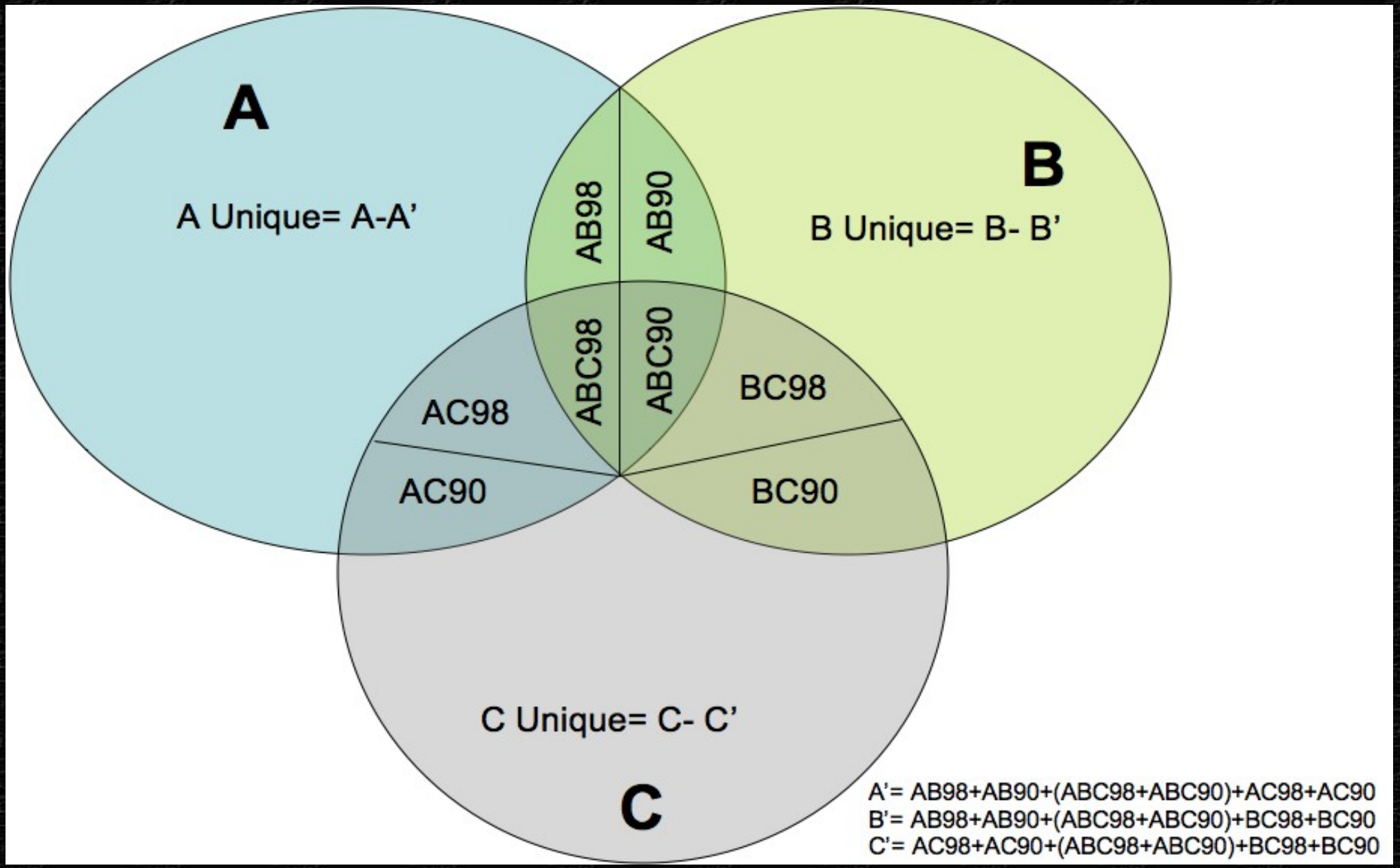
SeqLab.net + BioPerl + NCBI BLAST (simple task)

```
$ seqlab.pl --find_consensus \
   --forward TAGCTAGCTGGTTGGACGGATCGGATGAC \
   --reverse CCGATCCGTCAACCAGCTAGCTACGT

Query: 1-24 (of 29)  Hit: 4-26 (of 26)
tagctagctggttggacggatcgg
|||||||||||||| ||||||||||
tagctagctggtt-gacggatcgg

   tagctagctggttggacggatcggatgac
   |||||||||||||| ||||||||||
acgtagctagctggtt-gacggatcgg

acgTAGCTAGCTGGTTgGACGGATCGGatgac
```

SeqLab.net + BioPerl + NCBI BLAST (complex task)

- Given any arbitrary number of organisms, each containing an arbitrary number of sequences, discover all of the “strong”, “fairly strong”, and “weak” similarities between sequences. Output reports which tell us lots of information about your discoveries and create microarray specifications so we can fab a few thousand chips to test real-world samples against all proteins those organisms code for. We want to see whether or not Mr. Jones' illness may be related to these particular bugs.

Buckets...


A
A Unique= A-A'
B
B Unique= B- B'
AB98
AB90
ABC98
ABC90
AC98
AC90
BC98
BC90
C Unique= C- C'
C
A'= AB98+AB90+(ABC98+ABC90)+AC98+AC90
B'= AB98+AB90+(ABC98+ABC90)+BC98+BC90
C'= AC98+AC90+(ABC98+ABC90)+BC98+BC90

SeqLab.net + BioPerl + NCBI BLAST (complex task)

For 3 sequence sets (organisms), cross_blast() performs 6 different BLAST runs, accumulating results along the way:

"query" => "database"

ss1 => ss2
ss1 => ss3
ss2 => ss1
ss2 => ss3
ss3 => ss1
ss3 => ss2

SeqLab.net + BioPerl + NCBI BLAST (complex task)

Each of those BLAST runs requires...

- Walk through the query and database GenBank files. For each CDS, read the protein sequence, transform it to a FASTA file.
- Run the BLAST utility "formatdb" on the FASTA files.
- Run blastall on the files formatdb creates.
- Backtrack blast results (protein) in your original GenBank files to read the nucleotide sequences.

SeqLab performs all this drudgery for us. 6 times. Yay! (Imagine how much work we'd have to do manually if we had 5 organisms. Or 10. Yikes!)

SeqLab.net + BioPerl + NCBI BLAST (complex task)

SOLVED!

```
$ k.pl
```

:)

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

```
# slide 1 of 3
use SeqLab;

my $data_dir = "/home/jhannah/kiran/data";
my $data_tmp = "$data_dir/tmp";
unlink(glob "$data_tmp/*");

my $sl = SeqLab->new(storage => $data_tmp);

my $ss1 = $sl->new_SequenceSet(
   name => "Organism1"
);
$ss1->load(
   files => "$data_dir/ATCC12228_combo.gbk"
);
```

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

```
# slide 2 of 3
my $ss2 = $sl->new_SequenceSet(
   name => "Organism2"
);
$ss2->load(
   files => "$data_dir/SE_Org3.gbk"
);

my $ss3 = $sl->new_SequenceSet(
   name => "Organism3"
);
$ss3->load(
   files => "$data_dir/SE_RP62A_combo.gbk"
);
```

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

```
# slide 3 of 3
my ($stats, $H_pools) = $sl->cross_blast(
    type => "protein",
    SequenceSets => [ $ss1, $ss2, $ss3 ],
    hit_class_hierarchy => ['I', 'II', 'III'],
    hit_class_unique    => ['III'],
);
$stats->report1("report1_final.txt");
$stats->report2("report2_final.txt");

$sl->chipsets(
      SequenceSet => $ss1,   # I want Organism1
      stats       => $stats,
      output      => ".",
);
# All done!
```

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

Results

Running that program generates these files:

The .seq and .xls files are microarray definition data. The .txt files are reports.

```
ABC-I.Organism3.seq
ABC-I.Organism3.xls
ABC-II.Organism3.seq
ABC-II.Organism3.xls
AC-I.Organism3.seq
AC-I.Organism3.xls
AC-II.Organism3.seq
AC-II.Organism3.xls
BC-I.Organism3.seq
BC-I.Organism3.xls
C-unique.Organism3.seq
C-unique.Organism3.xls
report1_final.txt
report2_final.txt
```

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

Sample “Report 1”

```
Class I hits:
  Organism1                  Organism2                  Organism3
   27314465                    240997                    57636584
   27314478                  58003341                    57636554
27314531,27314539,27314550 \
                        13383307,13383313 \
                                  57635993,57636005,57636530,57638139
   27314683
                                                  57636374
Class II hits:
  Organism1                  Organism2                  Organism3
   27314792           (8101007),3201550                (57636783)

Class III hits:
  Organism1                  Organism2                  Organism3
 (27314764),(27316102)      (19172398)      (57636790),(57638035)
 (27315326)                   (886710)
                            (70987045)                 (57638504)
```

SeqLab.net cross_blast(), etc.

http://seqlab.net/pods2html/tutorial.html

Sample “Report 2”

Organism1

GI	I	II	III	bin	DeDupe	O_Pool
27315923	C		B	AC-I	0	1
27315719	C			AC-I	0	1
27315640	C		BC	AC-I	0	1
27314608			BC	A-unique	0	1
27315325		C	B	AC-II	0	1
27315149	Error: dupe of 27315137					

Organism2

GI	I	II	III	bin	DeDupe	O_Pool
37725698	A	C	AC	ABC-I	0	6
88683205	AC			ABC-I	0	15
27316891			AC	B-unique	0	1

Organism3

GI	I	II	III	bin	DeDupe	O_Pool
57636473			AB	C-unique	0	1
57637152	A		B	AC-I	0	1
57636398	B	AB		BC-I	0	1
57638147	Error: dupe of 57637502					

A GenBank file

```
LOCUS       AE015929     2499279 bp    DNA  circular BCT 04-JAN-2006
DEFINITION  Staphylococcus epidermidis ATCC 12228, complete genome.
ACCESSION   AE015929 AE016744 AE016745 AE016746 AE016747 AE016748
VERSION     AE015929.1  GI:27316888
AUTHORS     Zhang,Y.Q., Ren,S.X., Li,H.L., Wang,Y.X., Fu,G.
FEATURES    Location/Qualifiers
  CDS  15518..15847
    /product="conserved hypothetical protein"
    /protein_id="AAO03607.1"
    /db_xref="GI:27314470"
    /translation="MTTDLHTLVLIILCGVVTLLIRVIPFVMISRVNLPAIVIKWLSF
    IPITLFTALIIDGVIQQHDHAFGYTLNLPYIIAIVPTVMLAIFTRSLTVTILGGIFVI
    ACLRLIF"
ORIGIN
  1 aagaaattgt gacgcttatt tgaagttatc cacttataca cataatttct cgcaaaaatt
 61 gtggataaca catgcgctat acacacagtt attcaaaatt taacaacata ttcacagcca
121 tttgacatca cttggagtta aaaagtataa ttatgtggat aagtcgttca aattatgatt
181 ttacaaggat ttatttatta aatttatata cataaatggt gtgcataaat catagttatg
241 tttaagttat ccactgattg tgattaactt gtggataatt attaacatgc tgtgattatt
...
```

BioPerl a GenBank file

```
use strict;
use Bio::SeqIO;
my $seq_in = Bio::SeqIO->new(
  -file => "<$ARGV[0]", -format => "genbank"
);
while (my $inseq = $seq_in->next_seq) {
   my @features = $inseq->get_SeqFeatures(); # just top level
   foreach my $feat ( @features ) {
      next unless ($feat->primary_tag eq "CDS");
      my @db_xrefs = $feat->annotation->get_Annotations("db_xref");
      my @product =  $feat->annotation->get_Annotations("product");
      my @trans =    $feat->annotation->get_Annotations("translation");
      my $nucleic_seq = $feat->spliced_seq(-nosort => 1)->seq;
      my $protein_seq = $trans[0];
      @db_xrefs = grep { /^GI:/ } @db_xrefs;
      my $gi = $db_xrefs[0];
      $gi =~ s/^GI://;
      printf("locus_gi:%s", $inseq->primary_id);
      print " gi:$gi product:$product[0]\n";
      print "$nucleic_seq\n$protein_seq\n\n";
   }
}
```

BioPerl a GenBank file

```
$ perl j.pl genbank_sample.seq
locus_gi:1019382 gi:1019383 product:glutathione reductase (NADPH)
ATGACTTTTGATTATGACTTGTTTGTAATTGGTGCTGGTTCTGGTGGTTTGGCTGCTTCTAAACGAGCTGC
TAGCTATGGCGCAAAAGTAGCGATCGCCGAAAATGATTTAGTGGGTGGAACCTGTGTCATTCGGGGTTGTG
TACCCAAAAAACTCATGGTTTATGGTTCTCACTTTCCCGCTTTATTCGAGGATGCAGCAGGCTATGGTTGG
CAAGTCGGTAAGGCAGAATTAAATTGGGAACATTTCATTACATCTATAGATAAGGAAGTCCGGCGACTATC
CCAACTGCACATCAGCTTTCTAGAAAAAGCCGGGGTAGAACTGATCTCTGGTCGTGCTACTTTGGTAGATA
ATCACACAGTAGAAGTAGGCGAGCGTAAATTTACCGCCGATAAAATTTTAATTGCCGTTGGTGGTCGTCCC
ATCAAACCAGAGTTGCCAGGGATGGAATATGGCATCACCTCCAACGAAATTTTTCACCTAAAAACCCAACC
AAAACACATCGCTATCATTGGTTCTGGTTACATCGGTACAGAATTTGCCGGAATCATGCGTGGTTTGGGTT
CACAAGTCACCCAAATTACCAGAGGTGACAAAATTCTCAAAGGTTTTGATGAAGACATCCGCACCGAAATT
CAAGAAGGGATGACAAATCACGGTATTCGGATTATTCCTAAAAACGTAGTTACAGCTATTCAACAAGTACC
AGAAGGTTTGAAAATAAGTTTATCTGGTGAAGACCAAGAACCAATCATTGCCGATGTATTTTTAGTAGCTA
CAGGACGGGTTCCCAACGTAGATGGTTTAGGTCTGGAAAATGCTGGTGTTGATGTTGTTGACAGTTCTATA
GAGGGGCCAGGATACAGCACCATGAATGCCATTGCAGTGAACGAATACAGCCAAACCAGCCAACCCAATAT
CTATGCTGTTGGTGATGTTACAGACCGCTTAAACCTCACTCCCGTAGCCATTGGTGAAGGTCGCGCCTTCG
CCGACAGTGAATTTGGCAACAACCTCCGAGAATTTAGCCACGAAACTATTGCTACTGCTGTATTCTCTAAC
CCACAAGCCTCTACG
MTFDYDLFVIGAGSGGLAASKRAASYGAKVAIAENDLVGGTCVIRGCVPKKLMVYGSHFPALFEDAAGYGW
QVGKAELNWEHFITSIDKEVRRLSQLHISFLEKAGVELISGRATLVDNH
```

SeqLab::Pools

```
$ cat Pools.t
use Test::More tests => 52;

use SeqLab::Pools;

ok($p = SeqLab::Pools->new(),                    "new()");
ok($p->add(1,2),                                 "add()");
ok(my $pools = $p->get_pools,                    "get_pools()");
is_deeply($pools,  [ [ 1,2 ] ],                  "pools are as expected()");
ok($p->add(3,4),                                 "add()");
ok(my $pools = $p->get_pools,                    "get_pools()");
is_deeply($pools,  [ [ 1,2 ], [ 3,4 ] ],    "pools are as expected()");
ok($p->add(2,4),                                 "add()");
ok(my $pools = $p->get_pools,                    "get_pools()");
is_deeply($pools,  [ [ 1,2,3,4 ] ],              "pools are as expected()");
ok($p = SeqLab::Pools->new(),                    "new()");
ok($p->add(1,2),                                 "add()");
ok($p->add(2,3),                                 "add()");
ok($p->add(6,7),                                 "add()");
ok($p->add(8,9),                                 "add()");
ok($p->add(3,9),                                 "add()");
ok(my $pools = $p->get_pools,                    "get_pools()");
is_deeply($pools,  [ [1,2,3,8,9], [6,7] ],
                                                 "pools are as expected()");

...
```

SeqLab::Pools

```
$ perl Pools.t
1..52
ok 1 - new()
ok 2 - add()
ok 3 - get_pools()
ok 4 - pools are as expected()
ok 5 - add()
ok 6 - get_pools()
ok 7 - pools are as expected()
ok 8 - add()
ok 9 - get_pools()
ok 10 - pools are as expected()
ok 11 - new()
ok 12 - add()
ok 13 - add()
ok 14 - add()
ok 15 - add()
ok 16 - add()
ok 17 - get_pools()
ok 18 - pools are as expected()
...
```

Thank you!

Collaborative Laboratory for Applied Bioinformatics (CLAB)
http://clab.ist.unomaha.edu/CLAB/

Join us!

Omaha Perl Mongers
http://omaha.pm.org