MCB 5472
Computer Methods in Molecular Evolution
Spring 2014

Dr. J. Peter Gogarten jpgogarten@gmail.com
Dr. Jonathan Klassen jonathan.Klassen@uconn.edu
Email is good; please include "MCB 5472" in title

Outline

1. Pre-exam
2. Course syllabus (Klassen)
3. Student projects (Gogarten)
4. Intro to Unix & Perl (Klassen)
5. Exercises

Course Website

http://mcb5472.clas.uconn.edu

- All assignments and lectures will be posted here
- Comments are enabled, use these before emailing Jonathan and Peter directly
- Answering and discussing comments is encouraged, and can count towards your participation grade

Student Projects

- Should be related to your interests !!!
- Keep it simple -- this is a class project not a PhD thesis
- Examples for possible projects:

Evolution of a gene family

- When in the evolution of the interferon gene family (or whatever you are interested in) did gene duplications occur?
- Can one discriminate between gene conversion and multiple parallel transfers?
- Were the "gene duplications" additive transfers between close relatives?
- Which of the resulting subfamilies (if any) have acquired a new function?
- What is the phylogenetic distribution of this subfamily? (Would you expect members of this subfamily to be present in insects, fish, chicken, fungi, archaea?)
- Can you detect episodes of positive selection or of relaxed purifying selection?
- Is there anything that would suggest gene conversion events?

The " to-do-list" would include:

- gather data (note for some of the questions mentioned above you'll need aa and nucleotide sequences),
- align sequences
- build phylogenies
- analyze sequences
- assess reliability of branches
- INTERPRET WHAT YOU GOT!

Example: Can one detect a distinct divergence peak in the divergence between paralogs in putatively chimeric genomes?

Genome fusions are the latest rage in evolutionary biology:
For example:

- Koonin EV, Mushegian AR, Galperin MY, Walker DR. *Comparison of archaeal and bacterial genomes: computer analysis of protein sequences predicts novel functions and suggests a chimeric origin for the archaea.* Mol Microbiol. 1997 Aug;25(4):619-37.
- The Eukaryotes are a chimera of at least an archaeal like host cell and a bacterium that evolved into a mitochondrium (+ in some cases a cyanobacterium that evolved into a plastid)
- The Haloarchaea contain many bacterial genes
- The Thermotogales contain many archaeal genes
- Most plants and many fungi (likely including bakers yeast) are aneupolyploids

In most of these instances it is not clear that the transfer (or duplication) really occurred in a single massive event, or if the transfers (duplications) occurred on a gene by gene basis. (in yeast the type of genes that were duplicated suggest distinct selection pressures, see Benner et al http://www.sciencemag.org/content/296/5569/864.full)

Example: Chimera? continued

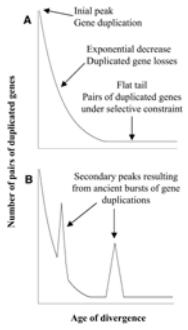
In case of a chimera formed in a single historic event one would expect

A) Two distinct types of phylogenetic affinity.

E.g.: Genes in *Thermotoga maritima* should either group with the sistergroup of the bacterial partner, or with the sistergroup of the archaeal donor.

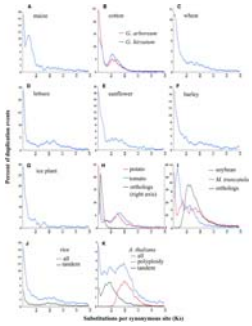
B) An ancient genome duplication or chimera formation should be revealed by peaks in the divergence of paralogs.

Theoretical Age Distributions of Pairs of Duplicated Genes in a Genome



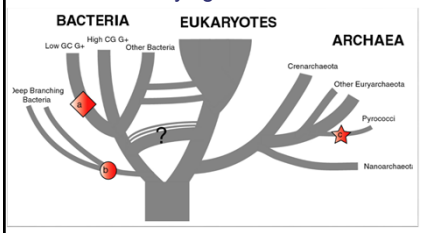
Blanc, G., et al. Plant Cell 2004;16:1667-1678

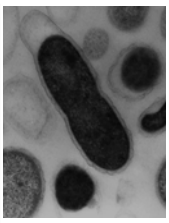
Distributions of the Fraction of Duplication Events as a Function of Their Levels of Synonymous Substitution for 14 Model Plant Species



Blanc, G., et al. Plant Cell 2004;16:1667-1678

The Phylogenetic Position of *Thermotoga*





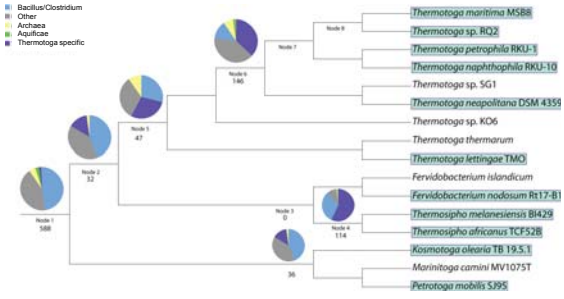
(a) Most genome based approaches
(b) according to 16S & other ribosomal markers
(c) according to phylogenetically discordant genes

Zhaxybayeva O, Fournier G, Lapierre P, Swithers K, Nesbø C, Gogarten JP, Doolittle WF, Noll KM: *Phylogenetic position of Thermotogales revisited: conflicting signals and evidence for thermophilic ancestral proteome*, PNAS 3/24/2009

Gophna U, Doolittle WF & Charlebois RL:
Weighted genome trees: refinements and applications. J. Bacteriol. (2005)

Gogarten JP & Townsend JP: Horizontal gene transfer, genome innovation, and evolution Nature Reviews in Microbiology 3(9) 679-687(2005)

Progressive gene gain in the Thermotogales



The acquisition of archaeal and clostridial genes appears to be an ongoing process.

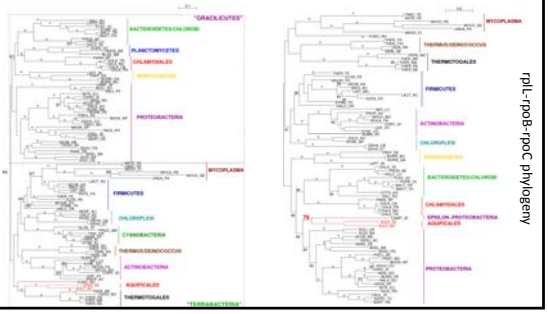
Research article

Highly accessed | Open Access

Accounting for horizontal gene transfers explains conflicting hypotheses regarding the position of aquificales in the phylogeny of Bacteria

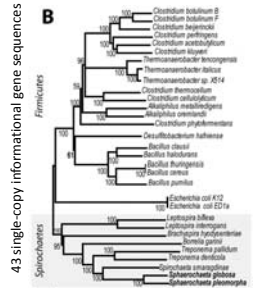
Bastien Boussau*, **Laurent Guéguen** and **Manolo Gouy**

BMC Evolutionary Biology 2008, **8**:272 doi:10.1186/1471-2148-8-272

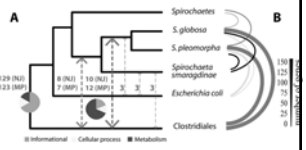


The Chimeric Genome of *Sphaerochaeta*: Nonspiral Spirochetes That Break with the Prevalent Dogma in Spirochete Biology

A. Caro-Quintero,[a] K. M. Ritalahti,[b,c] K. D. Cusick,[b] F. E. Löffler,[b,c,d] and K. T. Konstantinidis[a,e]



"The Sphaerochaeta genomes are highly enriched in fermentation and carbohydrate metabolism genes relative to other spirochetes, indicating a fermentative lifestyle. Remarkably, most of the enriched genes appear to have been acquired from non spirochetes, particularly clostridia, in several massive horizontal gene transfer events (>40% of the total number of genes in each genome)."



the values next to the arrows indicate the numbers of genes exchanged (out of a total of 178 genes examined).

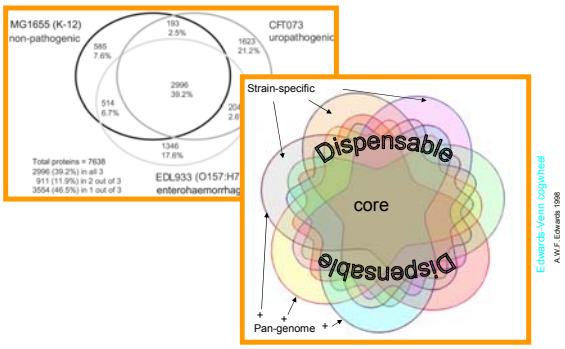
mBio

Chimera Example, continued

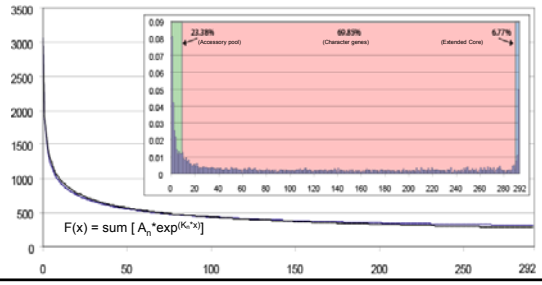
The " to-do-list" would include:

- Formulate the question you want to address
- Download and analyze the required genomes
- Run blastall (this might take a couple of hours)
- Analyze the results in an Excel spreadsheet
- Selected some genes (e.g., the ones that are most archaeal), assemble gene families and reconstruct their phylogenies.
- INTERPRET YOUR RESULTS! What does it all mean?

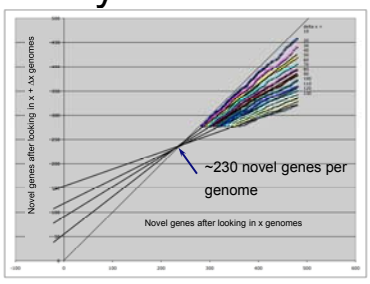
Example: Open or Closed Pan Genomes



Gene frequency in a typical genome



Kézdy-Swinbourne Plot



If f(x)=K+A • exp(-k•x), then f(x+Δx)=K+A • exp(-k•(x+Δx)).

Through elimination of A:

f(x+Δx)=exp(-k • Δx) • f(x) + K'

And for x→∞, f(x)→K, f(x+Δx)→K

only values with x ≥ 80 genomes were included

Even after comparing to a very large (infinite) number of bacterial genomes, on average, each new genome will contain about 230 genes that do not have a homolog in the other genomes.

- How does the size of the pan-genome differ for different group of organisms?
- For highly sampled organisms (such as E. coli) is the pan-genome open, or can one observe saturation at very high levels of sampling?
- Can one observe saturation for the bacterial pan-genome?
- The BBC bioinformatics facility maintains pan-genome collections for bacteria and archaea at the 90% and 95% 16S rRNA identity cut-off

Example: Screen for accessory genes in collections of mostly incomplete genome sequences.

- Sequencing genomes has become cheap, but many of the sequenced genomes are not closed. (Examples at UConn: *Halorubrum* and *Aeromonas* genomes.)
- Mapping read/contigs onto a closed reference genome, screen for genes absent in the reference genome
- In case of larger contigs, determine the location of the accessory genes
- Determine function these accessory genes.
- Where did these genes originate?

Example: How are molecular parasites distributed over the tree/net of life?

- Build position specific scoring matrices (PSI-BLAST) or Profiles for Hidden Markov Models (HMMER) for proteins that characterize (molecular) parasites (virus coat proteins, transposase, or integrase genes)
- Using the same collection of pan-genomes for groups with 5 and 10% 16S rRNA divergence, determine how many matches are in each of these group pan-genomes.
- Use other genome collections, including draft genomes as targets.

MCB 5742 Assignment #1: Introduction to the terminal and Perl January 22, 2014

Jonathan Klassen
jonathan.klassen@uconn.edu
(please put "MCB 5472" in your email title)

Outline

1. Introduction to UNIX
2. Logging on to the Biotechnology Center Cluster
3. Beginning with Perl

Introduction to UNIX

- Nearly all bioinformatics software runs on UNIX and its derivatives (e.g., LINUX and Mac OS)
- Very little bioinformatics software runs on Windows
- Bioinformatics is very strongly tied to the open-source software movement
  - Lots of help available on-line
  - Most programs are free
  - Windows is not very open-source friendly

Windows users:

- Option 1: Do all of your work connected to the Biotechnology Cluster server. Download sshclient (ftp://ftp.uconn.edu/restricted/ssh/)
- Option 1: Install LINUX to run in parallel with Windows (e.g., Biolinux http://nebc.nerc.ac.uk/tools/bio-linux)

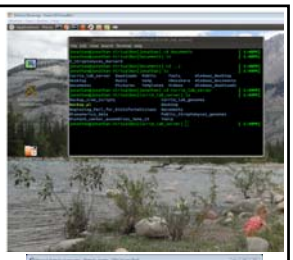
Unix and Perl Primer for Biologists

- The designated text for this course is "Unix and Perl Primer for Biologists", which can be found here: http://korflab.ucdavis.edu/Unix_and_Perl/unix_and_perl_v3.1.1.pdf

(Or on the website)

Terminal

- The terminal is the primary way to do computational biology
- Mac: Utilities/Applications/ Terminal
- Linux: Applications/Accessories/ Terminal
- Windows: sshclient



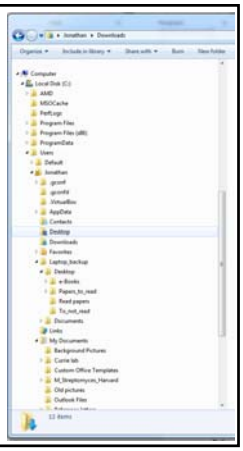
Terminal commands

- Using the terminal requires learning a series of commands that you will learn in the first exercise (Korf text U1-U28)
- These mainly involve doing various things to files
- These commands are extremely powerful once you learn them.

Directory trees

- All computer files are organized hierarchically
- Each folder has an address

/Users/Jonathan/
Laptop_backup/Destop/
e-Books



Navigation commands

`ls` – list directory contents

`cd` – change directory

```
cd Documents (move up to directory "Documents")
cd /Users/jlklassen/Documents (the same)
cd Documents/stuff (move up 2 directories)
cd /Users/jlklassen/Documents/stuff (the same)
cd ../ (move down 1 directory)
cd ../Desktop (move down 1 then up 1)
cd /Users/jlklassen/Desktop (the same)
cd ~/Desktop (the same)
```

More navigation commands

`cp` – copy documents & directories

```
cp stuff.stuff Desktop/ (make a copy in Desktop/)
cp stuff.stuff more_stuff.stuff (make a copy here)
```

`mv` – move documents & directories

```
mv Documents Desktop/ (move Documents into Desktop/)
mv Documents Desktop (rename Documents as Desktop)
```

`rm` – delete documents (BE CAREFUL)

```
rm to_delete.stuff (delete)
```

`mkdir` – make a new directory

```
mkdir New_dir (make new directory here)
mkdir Documents/New_dir (make new directory in New_dir)
```

`rmdir` – remove a directory

```
rmdir Old_dir (delete)
```

The Biotechnology Center cluster

- Your computer is good enough for basic tasks
- However, many tasks require more horsepower
- We connect to more powerful computers remotely using "ssh"
- Mac OS and Linux: follow directions in your email
- Windows: use sshclient

Working on the Cluster

- The cluster works by queuing all the jobs to be run and sending them to a free "node" as they become available.
- ssh'ing onto the server starts you all on the same "head node"
  - DO NOT EXECUTE PROGRAMS HERE
- For simple scripts, type "qlogin" and then run your scripts as normal
- For more involved jobs use the qsub system – see http://bbcsrv3.biotech.uconn.edu/wiki/index.php/Qsub

Text editors

- Mac OS: textwrangler
- Linux: gedit
- Windows: download gedit https://wiki.gnome.org/Apps/Gedit
- Cluster: `nano` or `vi`
- It is often easiest to write your script using a graphical text editor, move it to the cluster and then make only minor edits using nano or vi.

Some Unix tips

- Do not use spaces in your filenames
- Do not use a word processor instead of a text editor
- "`*`" is a wildcard character and is exceeding useful to match more than one thing
  - `mv *.pl Documents/`
  (move everything ending in ".pl" into the folder "Documents")
  - `mv *.p*l Documents/`
  (move everthing ending in ".p"+something or nothing+"l", i.e., both stuff.pl and stuff.perl would get moved)

More Unix tips

- Use ">" to redirect output from the screen into a file
```
perl script.pl > outfile.out
```
- Use "|" to redirect output into another program
```
perl script.pl | wc
```
- Use "`&`" to run program in the background
```
perl script.pl&
```
- Use "`nohup`" to run the program after you exit the terminal
```
nohup perl script.pl& > nohup.out
```
- File extensions can be whatever you want them to be, although conventions do exist (e.g., `.pl`)
  - This is a good way of keeping track what files contain

Perl

- Perl is a programing language that is great for scripting, i.e., tying programs together and reformatting their input and output
- By convention, perl scripts end with "`.pl`"
- Execute method #1:
```
perl script.pl
```
- Execute method #2:
```
chmod u+x script.pl
./script.pl
```

Perl rule #1

- For normal scripts, the first line of every script must be:

```
#!/usr/bin/perl
```

- For the cluster, however, this line must be:

```
#!/bin/env/perl
```

This tells the computer where to find the perl software itself

Perl rule #2

- Add comments to your code, otherwise neither me nor you will have any idea what it means the next time we look at it!
- Comments are everything that follows "#", except for the `#!/bin/env/perl` line

e.g., `# this is a comment`

e.g., `print "something"; # this part is a comment`

Perl rule #3

- Every line of code ends with a "`;`" character
- If you add an "enter" to your code, perl will keep reading your code until it reaches the next "`;`"

A command - `print`

- The `print` command…prints things!
  - By default to the screen
- Text to be printed must be in single or double quotation marks

```
print "This is text";
print 'This is text';
```

Interpolation

- Perl has fancy characters built into it
  - "`\t`" means print a tab character
  - "`\n`" means print a new line character (enter)
- Double quotation marks tell perl to treat these as single characters

```
print "This\tis a tab";
```

(outputs "This is a tab")

- Single quotation mark tell perl to treat these characters literally

```
print 'This\tis not a tab';
```

(outputs "This\tisnot a tab")

Strings

- Strings are most basic type of variable in perl
  - Think of them as words
- All string variable names start with a "$",

  e.g., `$string`
- Assign a string variable using "="

```
$string = "text";
print $string; outputs "text"
$string = 42;
print $string; outputs "42"
$string = "text \t text";
print $string; outputs "text      text"
```

i.e., interpolates

Math operators

• Perl can do all simple mathematical operations

```
$add = 1 + 2;
$subtract = 2 – 1;
$multiply = 2 * 2;
$divide = 5 / 3;
```

note: floating point numbers

```
$exponent = 2 ** 2;
$modulo = 3 % 2; # i.e., remainder
```

Comparison operators

- Greater than: >
- Less than: <
- Greater than or equal: >=
- Less than or equal: <=
- Equal (numeric): ==
- Not equal (numeric): `!=`
- Equal (string): `eq`
- Not equal (string): `ne`

Conditional statements

- A common task is to evaluate if something is true using if, elseif and else.
  - Note curly brackets denoting blocks do not need semicolons

```
if ($number < 4){
   print “$number is less than 4”;
}
elseif ($number == 4){
   print “$number is equal to 4”;
}
else {
   print “$number is greater than 4”;
}
```

Conditional statements 2

- Comparing strings:

```
if ($desert eq “chocolate”){
   print “Don’t share $desert”;
}
elseif ($desert ne “tapioca”){
   print “Share $desert”;
}
else {
   print “Pass on $desert”;
}
```

Opening an input file

- Use the `open` command
  - Specify a name for that file in the script, by convention in block capital letters
  - List file name in quotes

```
open (INFILE, “infile.in”);
```

  - “infile.in” is now recognized by perl as “INFILE”
- Pro tip: tell your script to exit if it cannot open in input file

```
open (INFILE, “infile.in”) or
     die “Cannot open infile.in”;
```

  - `die` tells the program to exit and print out a message why it died

Opening an output file

- Use the same open command, but with one important difference:

```
open (OUTFILE1, “>outfile1.out”) or
     die “Can’t open outfile1.out”;
```

  - One “>” will overwrite outfile1.out if it already exists

```
open (OUTFILE2, “>>outfile2.out”) or
     die “Can’t open outfile2.out”;
```

  - Two “>>”s will append new output to the and of outfile2.out

Reading through your file

- A simple way to read an input file is using while loop combined with a special input operator

```
while ($line = <INFILE>){
    print $line;
}
```

  - In words:
    (1) Read `INFILE` line by line
    (2) Assign each line to the string `$line`
    (3) Print that line (`$line`) to the screen
    (4) Stop when you run out of lines

Printing to an output file

- Make sure an output file was opened first!
- Use the print command:

```
print OUTFILE $line;
```

  print `$line` to the `OUTFILE`

```
print $line;
```

  prints `$line` to the screen
- Your output file stays open until you close it

```
close OUTFILE;
```

Assignment #1:

- View on the course website: http://mcb5472.clas.uconn.edu
- Complete each task, and email the output, scripts, and input files to Jonathan (jonathan.klassen@uconn.edu; include “MCB 5472” in title)