Array and hash are VERY useful to hold text data in the memory and are often created using loops

- Creating Array:

```
@fruit_list = ('apple', 'orange', 'banana');
```

=

```
$fruit_list[0]='apple';
$fruit_list[1]='orange';
$fruit_list[2]='banana';
```

```
@a=<>;
```

←

```
apple
orange
banana
```

- Creating Hash:

```
%ip2hostname = (
  "glu" => "131.156.41.196",
  "gly" => "131.156.41.193",
  "cys" => "131.156.41.195"
);
```

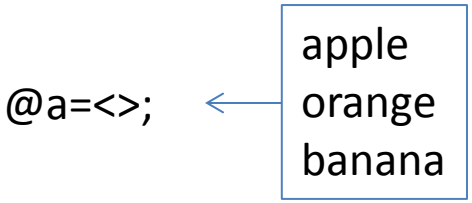
=

```
$ip2hostname{"glu"}="131.156.41.196";
$ip2hostname{"gly"}="131.156.41.193";
$ip2hostname{"cys"}="131.156.41.195";
```

```
$hash{$col[0]}=$col[1];
```

←

glu	131.156.41.196
gly	131.156.41.193
cys	131.156.41.195



push

A perl function to create an array by pushing an element into a list from behind

How?

push(@array, $element);

In a loop, push $element to an array to create it

Asia	China
Asia	Japan
Asia	India
Europe	UK
Europe	France
Europe	Germany
North America	US
North America	Canada
North America	Mexico

$col[0] **$col[1]**

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

while(<>){
  @col=split(/\t/,$_);
  push(@country,$col[1]);
}

print @country;
```

China
China, Japan
China, Japan, India
China, Japan, India, UK
China, Japan, India, UK, France

…

China, Japan, India, UK, France, Germany, US, Canada, Medixo

This is the same as the cut –f2 command of Shell

What if we want to group the countries based on the continent?
We will have three separate arrays: @Asia, @Europe, @North_America

```
@Asia=("China", "Japan", "India");
@Europe=("UK", "France", "Germany");
@North_America=("US", "Canada", "Mexico");
```

How can we hold both continent and country info using a variable?

Hash of array is one of the complex data structures built on top of basic ones (scalar, array and hash), designed for one-to-many data tables

```
%country = (
        "Asia" => ["China", "Japan", "India"],
        "Europe" => ["UK", "France", "Germany"],
        "North America" => ["US", "Canada", "Mexico"]
);
```

=

```
$country{"Asia"} = ["China", "Japan", "India"];
$country{"Europe"} = ["UK", "France", "Germany"];
$country{"North America"} = ["US", "Canada", "Mexico"];
```

How do we create a hash of array from a tabular file?

Asia	China
Asia	Japan
Asia	India
Europe	UK
Europe	France
Europe	Germany
North America	US
North America	Canada
North America	Mexico

Hash of array: %country

Key	**Value is an array**
"Asia"	("China", "Japan", "India")
"Europe"	("UK", "France", "Germany")
"North America"	("US", "Canada", "Mexico")

We need to change the array to a hash of array by adding the KEY part:

```
push(@country,$col[1]);

@{$country{KEY}}
```

$col[0]	$col[1]
Asia	China
Asia	Japan
Asia	India
Europe	UK
Europe	France
Europe	Germany
North America	US
North America	Canada
North America	Mexico

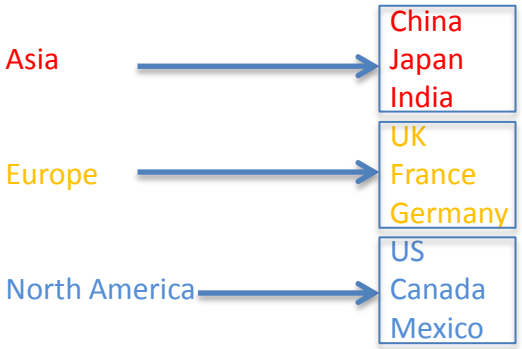
$country{$col[0]}=$col[1];

```
While(<>){
    @col=split(/\t/,$_);
    push(@{$country{$col[0]}},$col[1]);
}
```

value

key

Curley for the hash name part



```
#!/usr/bin/perl

open(IN,$ARGV[0]);

while(<IN>){
  @col=split(/\t/,$_);
  push(@{$country{$col[0]}},$col[1]);
}

foreach $continent(keys %country){
  @country=@{$country{$continent}};  # for each key, assign the value to an
array @country
  print "###$continent has the following countries\n";
  foreach(@country){
    print $_,"\n";
  }
}
```

vi country.pl

Create hash of array

Print hash of array

```
Asia        China
Asia        Japan
Asia        India
Europe      UK
Europe      France
Europe      Germany
North America       US
North America       Canada
North America       Mexico
```

vi country

perl country.pl country

1	Hit-ID	6	i-E-values
2	Hit-length	7	HMM-start
3	HMM-length	8	HMM-end
4	Full length E-values	9	Hit-start
5	c-E-values	10	Hit-end

```
yyin@glu:~/work/class$ less GH5.hmm.cowrumen.dm.ps| head
NODE_457020_length_97146_cov_14.955994_orf_01700     782     275     2.9e-71 1.2e-45 8.1e-43 2       239     68      328
NODE_457020_length_97146_cov_14.955994_orf_01700     782     275     2.9e-71 2.3e-28 1.5e-25 7       228     409     651
NODE_2314521_length_30819_cov_0.660826_orf_30190     715     275     3.3e-55 4.7e-32 3.1e-29 4       243     41      301
NODE_2314521_length_30819_cov_0.660826_orf_30190     715     275     3.3e-55 6.9e-26 4.5e-23 2       239     344     601
NODE_457020_length_97146_cov_14.955994_orf_01790     995     275     1.7e-54 6.3e-29 4.1e-26 2       237     41      311
NODE_457020_length_97146_cov_14.955994_orf_01790     995     275     1.7e-54 1.1e-27 7e-25   2       240     358     625
NODE_4002281_length_100204_cov_2.154804_orf_16350    624     275     2.2e-52 5.3e-55 3.4e-52 1       239     180     522
NODE_4159482_length_690369_cov_12.576118_orf_32020   762     275     1.6e-51 4.2e-54 2.7e-51 1       239     371     711
NODE_2008511_length_308405_cov_4.203573_orf_06380    745     275     3.7e-50 9.5e-53 6.2e-50 2       240     47      433
NODE_459624_length_630037_cov_23.694723_orf_16430    676     275     5.6e-50 1.2e-52 8e-50   2       241     283     625
```

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

open(ID,$ARGV[1]);
while(<ID>){
  chomp $_;

                    @col=split(/\t/,$_);
                    push(@{$id_hash{$col[0]}},$_);

}
```

Load the tabular file into memory

1) Cut the tabular file and create a hash with ID as the key and the entire row as the values
2) Since there are two lines with the same ID, so hash of array will use the ID as key and both lines as value (an array with two elements)

Find out which IDs have multiple GH5 domains

```
less GH5.hmm.cowrumen.dm.ps| cut -f1 | sort | uniq
-c | awk '$1!=1'
```

vi hash_of_array.pl

```
#!/usr/bin/perl

open(IN,$ARGV[0]);

while(<IN>){
  chomp $_;
  @col=split(/\t/,$_);
  push(@{$id_hash{$col[0]}},$_);
}

foreach $key(keys %id_hash){
  @id_array=@{$id_hash{$key}};  # for each key, assign the value to an array
  if($#id_array>=1){ # $#id_array returns the index of the last element
    print "###this is the array with the key: $key\n";
    foreach(@id_array){
      print $_,"\n";
    }
  }
}
```

Create a hash of array as explained in the above example

Print elements with >=2 lines in the array

```
perl hash_of_array.pl GH5.hmm.cowrumen.dm.ps
```

```
#!/usr/bin/perl

use Bio::SeqIO;

open(IN,$ARGV[0]);

while(<IN>){
  chomp $_;
  @col=split(/\t/,$_);
  push(@{$id_hash{$col[0]}},$_);
}

$new=Bio::SeqIO->new(-file=>$ARGV[1], -format=>"fasta");

while($seq=$new->next_seq){
  if(defined $id_hash{$seq->id}){
   @id_array=@{$id_hash{$seq->id}};

   foreach(@id_array){
     @id_col=split(/\t/,$_);
     print ">",$id_col[0],"|",$id_col[-2],"-",$id_col[-1],"\n",
     $seq->subseq($id_col[-2],$id_col[-1])."\n";
   }

  }
}
```

vi get-subseq-bioperl.pl

perl get-subseq-bioperl.pl GH5.hmm.cowrumen.dm.ps metagenemark_predictions.faa | less

Create a hash of array as explained in the above example

If the ID is defined in the %id_hash, create a temporary array to hold the lines of existing IDs (it's ok if there is only one line in the array)

subseq method is called to help retrieve the domain segments using the position info

Build the fasta description line in a format: >ID|start-end

Perl one-liner

You don't write codes into a file and then issue "`perl file.pl`" on the command line;
You write the codes directly on the command line, like you are typing regular Linux commands

```
#!/usr/bin/perl

open (IN,$ARGV[0]);

@a=<IN>;

foreach(@a){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

=

```
#!/usr/bin/perl

while (<>){
   @col=split(/\t/,$_);
   print $col[1],"\tmutation\n";
}
```

=

```
perl -e 'while(<>){@col=split(/\t/,$_);print $col[1],"\tmutation\n";}' cosmicRaw.txt.head10.6col
```

=

```
cat cosmicRaw.txt.head10.6col | cut -f2 | awk '{print $1,"mutation"}' | sed 's/ /\t/'
```

```
seqret -sequence /media/DATAPART4/z1003529/sequence.gp -outseq sequence.gp.fa -
sformat genbank -osformat fasta
```

vi format-bioperl.pl

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

use Bio::SeqIO;

$in=Bio::SeqIO->new(-file=>$ARGV[0],-format=>$ARGV[1]);
$out=Bio::SeqIO->new(-file=>">$ARGV[2]",-format=>$ARGV[3]);

while ($seq = $in->next_seq() ) {
  $out->write_seq($seq);
}
```

```
perl format-bioperl.pl /media/DATAPART4/z1003529/sequence.gp genbank
sequence.gp.fa fasta

perl format-bioperl.pl /media/DATAPART4/z1003529/sequence.gp genbank
sequence.gp.fa fasta 2>err
```

```
yyin@glu:~/work/class$ ls /usr/share/perl5/Bio/SeqIO
SeqIO/     SeqIO.pm
yyin@glu:~/work/class$ ls /usr/share/perl5/Bio/SeqIO
abi.pm       chadoxml.pm     excel.pm            game.pm       kegg.pm       MultiFile.pm  scf.pm        tigr.pm
ace.pm       chaos.pm        exp.pm              gbdriver.pm   largefasta.pm nexml.pm      seqxml.pm     tigrxml.pm
agave.pm     chaosxml.pm     fasta.pm            gbxml.pm      lasergene.pm  phd.pm        strider.pm    tinyseq
alf.pm       ctf.pm          fastq.pm            gcg.pm        locuslink.pm  pir.pm        swissdriver.pm tinyseq.pm
asciitree.pm embldriver.pm   flybase_chadoxml.pm genbank.pm    mbsout.pm     pln.pm        swiss.pm      ztr.pm
bsml.pm      embl.pm         FTHelper.pm         Handler       metafasta.pm  qual.pm       table.pm
bsml_sax.pm  entrezgene.pm   game                interpro.pm   msout.pm      raw.pm        tab.pm
yyin@glu:~/work/class$
```

How to use bioperl to retrieve GenBank entries remotely?

perldoc Bio::DB::GenBank

vi get-genbank.pl

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

use Bio::DB::GenBank;
use Bio::SeqIO;

$gb = new Bio::DB::GenBank;

$seqout = new Bio::SeqIO(-fh => \*STDOUT, -format => "genbank");

while(<>){
  chomp $_;
  $seq = $gb->get_Seq_by_id($_);
  $seqout->write_seq($seq);
}
```

```
cat /home/yyin/work/class/all.genpept.id | wc -l
138732
cat /home/yyin/work/class/all.genpept.id | perl get-genbank.pl
```

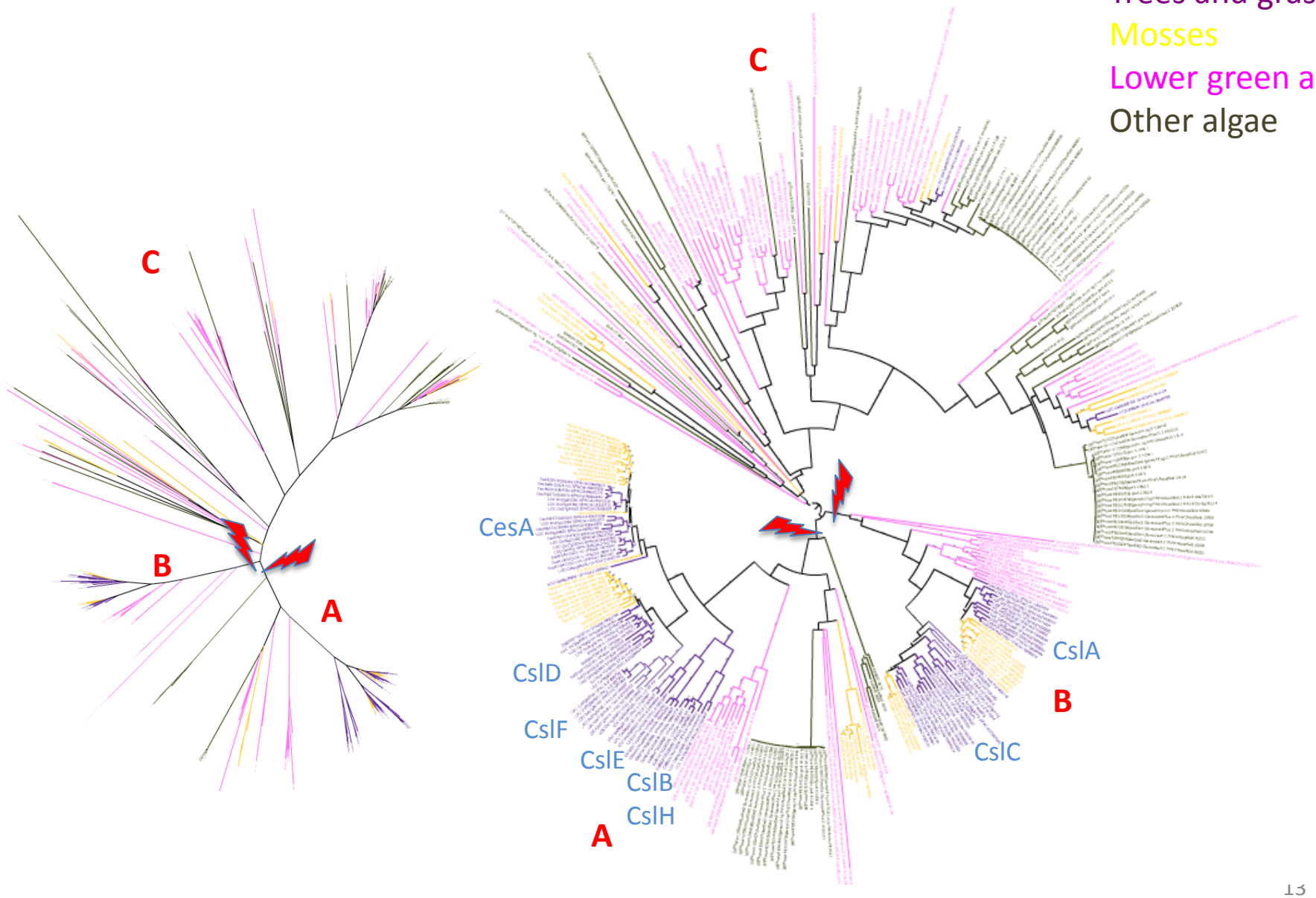
Another module that can do the same thing

vi get-genbank2.pl

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

use Bio::DB::EUtilities;

while(<>){

  $factory = Bio::DB::EUtilities->new(-eutil => "efetch",-db =>
"protein",-rettype => "gb",-email => "yanbin.yin\@gmail.com",-id => $_);

  print $factory->get_Response->content;

}
```

```
cat /home/yyin/work/class/all.genpept.id | perl get-genbank2.pl
```

How to parse newick format tree text file to retrieve IDs?



yyin@glu:~/work/class$ less full-genom.sel.fa.l.fasttree.nwk.2-sub1

Newick format: not human-friendly

```
yyin@glu:~/work/class$ less full-genom.sel.fa.l.fasttree.nwk.2-sub1
(((jgi|Ost9901_3|25890|estExt_fgenesh1_pg.C_Chr_200213:0.15902,jgi|OstRCC809_2|61153|fgenesh1_pm.chr_20_#_228
:0.27533):1.83361[1.00],(((g6025.t1|PACid-26893837:0.0,g6025.t2|PACid-26893838:0.0):0.13115,Vocar20003015m|PA
Cid-23131228:0.11793):0.96427[1.00],(((jgi|ChlNC64A_1|139598|IGS.gm_25_00149:0.40939,(jgi|Coc_C169_1|2833|gw1
.7.154.1:0.14667,(jgi|Astpho1|33644|e_gw1.00149.81.1:0.15812,jgi|Coc_C169_1|66059|estExt_Genemark1.C_70365:0.
15753):0.09256[0.98]):0.13253[0.99]):0.33178[1.00],((jgi|OstRCC809_2|60557|fgenesh1_pm.chr_16_#_137:0.25330,(
jgi|Ost9901_3|18489|fgenesh1_pg.C_Chr_16000054:0.18305,jgi|Ostta4|9161|fgenesh1_pm.C_Chr_17.0001000010:0.1944
7):0.04122[0.41]):0.33634[1.00],(jgi|MicpuC3|23128|MicpuC2.e_gw1.16.71.1:0.19723,jgi|MicpuN3|97997|fgenesh2_p
m.C_Chr_10000079:0.30466):0.04300[0.26]):0.26563[1.00]):0.24381[0.89],((CslC6|AT3G07330.1|PACid-19663909:0.25
950,((CslC12|AT4G07960.1|PACid-19644865:0.18820,((LOC_Os07g03260.1|PACid-21899603:0.09811,LOC_Os03g56060.1|PA
Cid-21911464:0.03241):0.08245[0.99],(LOC_Os01g56130.1|PACid-21908220:0.07294,LOC_Os05g43530.1|PACid-21939856:
0.05842):0.09024[1.00]):0.07664[0.97]):0.10087[1.00],(((140200|PACid-15405032:0.18777,442658|PACid-15412386:0
.13170):0.05478[0.93],(((Pp1s89_286V6.1|PACid-18046098:0.02931,Pp1s224_44V6.1|PACid-18053335:0.04428):0.15389
[1.00],(Pp1s162_130V6.1|PACid-18043543:0.06375,(Pp1s373_28V6.1|PACid-18059424:0.04625,Pp1s19_258V6.1|PACid-18
063508:0.04614):0.04348[0.99]):0.07078[1.00]):0.06561[0.98],(Pp1s164_5V6.1|PACid-18038263:0.03271,Pp1s15_115V
6.1|PACid-18050525:0.02650):0.20137[1.00]):0.07517[0.99]):0.08272[0.99],(CslC4|AT3G28180.1|PACid-19662684:0.2
4547,(LOC_Os08g15420.1|PACid-21888437:0.21486,(LOC_Os09g25900.1|PACid-21926657:0.22883,(CslC8|AT2G24630.1|PAC
id-19638556:0.07641,CslC5|AT4G31590.1|PACid-19646979:0.03274):0.11628[1.00]):0.04293[0.44]):0.03353[0.45]):0.
06688[0.99]):0.05248[0.61]):0.11373[0.65]):0.61724[1.00],(LOC_Os09g39920.1|PACid-21925118:0.71591,((LOC_Os08g
33740.1|PACid-21888939:0.25395,LOC_Os02g51060.1|PACid-21924463:0.47471):0.06092[0.14],((LOC_Os06g12460.1|PACi
d-21930248:0.37581,((LOC_Os07g43710.1|PACid-21901112:0.17583,LOC_Os03g26044.1|PACid-21914244:0.15585):0.30519
[1.00],(LOC_Os10g26630.1|PACid-21882652:0.16652,(LOC_Os09g26770.2|PACid-21927293:0.56424,LOC_Os03g07350.1|PAC
id-21911250:0.00015):0.08626[1.00]):0.16088[1.00]):0.09931[0.96]):0.13144[1.00],(((CslA9|AT5G03760.1|PACid-19
669977:0.16676,(CslA2|AT5G22740.1|PACid-19673036:0.21878,(LOC_Os02g09930.1|PACid-21919891:0.15886,LOC_Os06g42
020.1|PACid-21932621:0.15291):0.07181[0.85]):0.06807[0.75]):0.04907[0.78],(230176|PACid-15411642:0.23068,(Pp1
s36_214V6.1|PACid-18053528:0.04313,(Pp1s36_62V6.1|PACid-18053619:0.03188,Pp1s65_194V6.1|PACid-18057777:0.0375
7):0.01312[0.49]):0.29524[1.00]):0.07358[0.94]):0.03123[0.02],(CslA7|AT2G35650.1|PACid-19639665:0.34578,((CslA1|AT4G16590.1|PACid-19644242:0.23037,(CslA10|AT1G24070.1|PACid-19656371:0.14229,(CslA15|AT4G13410.1|PACid-19645825:0.13816,CslA11|AT5G16190.1|PACid-19666574:0.10137):0.02807[0.41]):0.05314[0.89]):0.21219[1.00],(CslA3|AT1G23480.1|PACid-19653790:0.13931,CslA14|AT3G56000.1|PACid-19665013:0.35565):0.09725[0.98]):0.09176[0.98]):0.08783[0.98]):0.11718[1.00]):0.05389[0.15]):0.08084[0.10]):0.41999[1.00]):0.24745[0.98]):0.31142[0.51]):0.78697[0.99]):0.20304[0.50],ConsensusfromContig37575-snap_masked-ConsensusfromContig37575-abinit-gene-0.2-mRNA-1-cds-7119/2753-2927-0-+:3.26499);
```

perldoc Bio::TreeIO

vi get-treeid.pl

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

use Bio::TreeIO;

$treeio=Bio::TreeIO->new(-format=>"newick", -
file=>$ARGV[0]);

while($tree=$treeio->next_tree){
  for $node ($tree->get_nodes){
    print $node->id."\n";
  }
}
```

```
perl get-treeid.pl /home/yyin/work/class/full-genom.sel.fa.l.fasttree.nwk.2-sub1
| less

perl get-treeid.pl /home/yyin/work/class/full-genom.sel.fa.l.fasttree.nwk.2-sub1
| sed '/^$/d' | less

perl get-treeid.pl /home/yyin/work/class/full-genom.sel.fa.l.fasttree.nwk.2-sub1
| sed '/^$/d' > sub1.id
```

How do we join two tabular files according to a common column?

Consider this task: find out how many cancer-related genes are transcription regulators?

The cancer Gene Census

http://www.nature.com/nrc/journal/v4/n3/abs/nrc1299.html

http://cancer.sanger.ac.uk/cancergenome/projects/census/
Download the excel sheet and open it
Copy and paste to create a file
/home/yyin/work/class/cancer-census

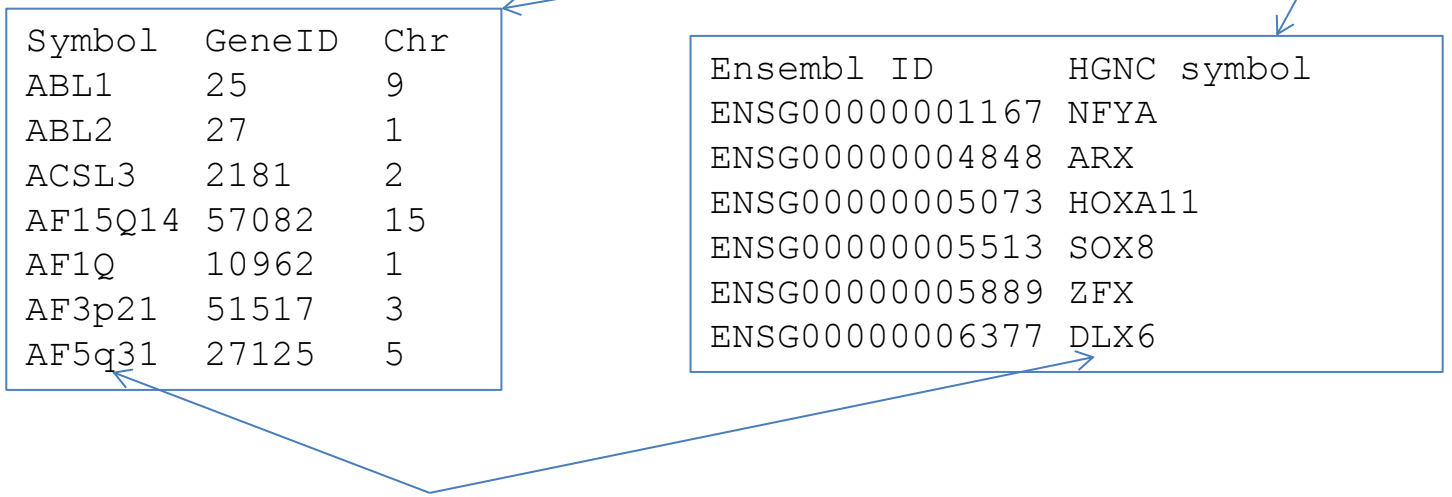
A census of Human Transcription Factors

http://www.nature.com/nrg/journal/v10/n4/suppinfo/nrg2538.html
Wget the Supplementary information S3
/home/yyin/work/class/nrg2538-s3.txt

What we need to do is to join the two tables together based on common col

First let's clean the two tables first:

```
cat nrg2538-s3.txt | cut -f2,6 > nrg2538-s3.txt.cut
less cancer-census | cut -f1,3,4 > cancer-census.cut
```



Symbol	GeneID	Chr
ABL1	25	9
ABL2	27	1
ACSL3	2181	2
AF15Q14	57082	15
AF1Q	10962	1
AF3p21	51517	3
AF5q31	27125	5

Ensembl ID	HGNC symbol
ENSG00000001167	NFYA
ENSG00000004848	ARX
ENSG00000005073	HOXA11
ENSG00000005513	SOX8
ENSG00000005889	ZFX
ENSG00000006377	DLX6

Common col is

Now we need a perl script to join the two tables:
using the gene symbol as the key to create a hash

vi join.pl

```
#!/usr/bin/perl

open(IN,$ARGV[0]);
open(IN2,$ARGV[1]);

while(<IN>){
  chomp $_;
  @col=split(/\t/,$_);
  $hash{$col[1]}=$_;
}

while(<IN2>){
  chomp $_;
  @col2=split(/\t/,$_);
  if(defined $hash{$col2[0]}){
    print $_,"\t",$hash{$col2[0]},"\n";
  }
  else{
    print $_,"\n";
  }
}
```

```
perl join.pl nrg2538-s3.txt.cut cancer-census.cut | less
perl join.pl nrg2538-s3.txt.cut cancer-census.cut | awk '$4!=""' | less
perl join.pl nrg2538-s3.txt.cut cancer-census.cut | awk '$4!=""' | wc -l
```

Use bioperl to extract sequences given IDs (get-seq-bioperl.pl)

Use bioperl to extract sub-sequences given tabular file with positions (get-subseq-bioperl.pl)

Given GenBank IDs, get fasta or genbank format from NCBI remotely (get-genbank.pl)

Bioperl to parse newick trees (get-treeid.pl)

Connecting two ids file to find common or different things (join.pl)

http://www.biogem.org/downloads/notes/BioPerl.pdf