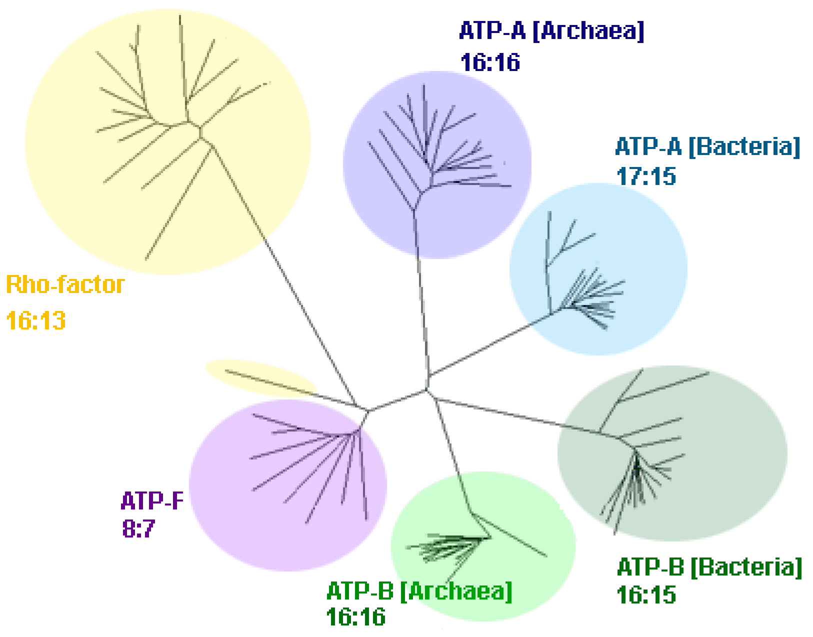
BranchClust

A Phylogenetic Algorithm for Selecting Gene Families

Version 1.01

Tutorial



http://www.bioinformatics.org/branchclust

Tutorial

Selection of orthologous families with BranchClust is a multi-stage process that includes processing BLAST results, assembling superfamilies, alignment, reconstruction of phylogenetic trees, and applying the BranchClust method itself. Here we describe step-by-step procedures used for selection of orthologous families for a set of different taxa. All perl scripts described here are freely available and can be downloaded as one archive from the web-site:

http://www.bioinformatics.org/branchclust/

All following procedures could be divided into 6 major stages:

I. Downloading complete genomes and collection of significant hits
II. Creation of taxa identification table
III. Assemblage of superfamilies
IV. Reconstruction of superfamily trees.
V. Selection of orthologous families with BranchClust
VI. Visualization of results with TreeDyn

For all the programs described below, it is assumed that they are run from inside one directory. Perl-scripts create output files that are further used as input by other programs so that each step should be performed in order described in the tutorial.

The scripts and programs were tested under both Mac OSX Darwin and Debian Linux operating sytems. Modules or standalone programs such as BioPerl, blastall, clustalw and treedyn need to be installed in the system. These programs are available from

BioPerl – http://www.bioperl.org
Blastall - http://www.ncbi.nlm.nih.gov/BLAST/download.shtml
Clustalw - http://bips.u-strasbg.fr/fr/Documentation/ClustalX/
TreeDyn - http://www.treedyn.org

I. Downloading complete genomes and collection of significant hits

1. Download files in fasta format (*.faa) with genes represented as amino acid sequences from NCBI web site:

ftp://ftp.ncbi.nlm.nih.gov/genomes/Bacteria

```
Result: fasta/*.faa
```

Note! Some bacterial genomes consist of more than one chromosome. To include all chromosomes in the analysis, download all *.faa files from species directory and put them in one file.

Put all genomes in `fasta` directory. All files must have an extension *.faa.

EXAMPLE: 13 gamma proteobacteria

```
1. Buchnera aphidicola str. Bp (Baizongia pistaciae)
2. Escherichia coli CFT073
3. Haemophilus influenzae Rd KW20
4. Pasteurella multocida subsp. multocida str. Pm70
5. Pseudomonas aeruginosa
6. Salmonella typhimurium
7. Vibrio cholerae O1 biovar eltor str. N16961
8. Wigglesworthia glossinidia endosymbiont of Glossina brevipalpis
9. Xanthomonas campestris pv. campestris str. ATCC 33913
10. Xanthomonas axonopodis pv. citri str. 306
11. Xylella fastidiosa 9a5c
12. Yersinia pestis CO92
13. Yersinia pestis KIM
```

Put all genomes in `fasta` directory, rename them at your own convenience.

```
>ls fasta
buch.faa        pasteur.faa     vcholera.faa    xancit.faa      ypkim.faa
ecoli.faa       pseudo.faa      wiggle.faa      xylella.faa
hinflue.faa     salmon.faa      xancam.faa      ypco.faa
```

2. Create database consisting of all genomes:

```
> perl create_one_faa.pl
```

```
Result: fasta_all/allgenomes.faa
```

EXAMPLE: 13 gamma proteobacteria

Database `fasta_all/allgenomes.faa` composed of 13 genomes contains 43 694 genes

3. Format database for use with BLAST:

```
> perl format_faa.pl
```

4. Blast database composed of all genomes against itself:

```
> perl do_blast.pl
```

Note! You can modify Blast parameters at your own convenience (see `blastall` string in the script). By default, E-value=e-4, tabular format.

```
Output: blast/all_vs_all.out
```

EXAMPLE: 13 gamma proteobacteria

Blast result file `blast/all_vs_all.out` has 1 759 999 hits.

5. Parse Blast results:

```
> perl parse_blast.pl
```

```
Input: blast/all_vs_all.out
Output: parsed/all_vs_all.parsed
```

Script requires BioPerl module. In the script `parse_blast.pl` replace path to the library with your path or contact system administrator and install BioPerl module in the system.

Script takes Blast results and creates output file in such a way that each gene from the input database with all the hits collected from the database including genome of the query is located in one line.

EXAMPLE: 13 gamma proteobacteria

After processing raw Blast output the file `parsed/all_vs_all.parsed` has 43 686 lines.

II. Creation of taxa identification table gi_numbers.out

All genome files in fasta format should be located in `fasta` directory.

1. Create taxa identification table:

```
>perl extract_gi_numbers.pl
```

```
Result: gi_numbers.out
Additional: gi_numbers.log
```

Each taxa is identified by gi-number template. Script automatically takes care of double-entries and increases resolution if necessary.

EXAMPLE: 13 gamma proteobacteria

```
>more gi_numbers.out

Buchnera aphidicola str. Bp (Baizongia pistaciae) |     2790....
Escherichia coli CFT073 |       2624....        2625....
Haemophilus influenzae Rd KW20 |        1627....        3099....        1867....
Pasteurella  multocida  subsp.  multocida  str.  Pm70  |        15602...            15603...
156019..          1560187.    1560188.          1560189.          15601866          15601867
15601868        15601869
Pseudomonas aeruginosa PAO1 |   1559....         156000..         156001..          156002..
156003..    156004..          156005..          156006..          1560070.          1560071.
1560072.        1560073.   1560074.        1560075.        1560076.
Salmonella typhimurium LT2 |    1676....         3954....         3998....          1723....
3244....
Vibrio cholerae O1 biovar eltor str. N16961 |   1564....         156010..           156011..
156012..    156013..          156014..          156015..          156016..          156017..
1560180.          1560181.    1560182.          1560183.          1560184.          1560185.
15601860          15601861          15601862    15601863          15601864          156008..
156009..        1560077.        1560078.        1560079.
Wigglesworthia  glossinidia  endosymbiont  of  Glossina  brevipalpis  |             5047....
3249....
Xanthomonas campestris pv. campestris str. ATCC 33913 | 2122....        2123....
Xanthomonas axonopodis pv. citri str. 306 |     2124....        2126....
Xylella fastidiosa 9a5c |       1583....        1095....        1542....
Yersinia pestis CO92 |  1608....        4054....        4078....        1612....
Yersinia pestis KIM |   2212....
```

III. Assemblage of superfamilies

1. Assemble Blast hits in superfamilies:

```
> perl parse_superfamilies.pl <NUMBER_OF_TAXA>
```

```
Result: parsed/all_vs_all.fam
Intermidiate result: parsed/all_vs_all.temp
```

First it filters Blast results so that each line contains at least `<NUMBER_OF_TAXA>` different species: results are saved in the intermediate file `parsed/all_vs_all.temp`. At the second stage it combines different sets according to the following rules.

Each line after parsing Blast output (file `parsed/all_vs_all.parsed`) contains all significant hits for a given query gene in all genomes including the genome of the query. It could happen (and it does happen) that another gene from that line, when used as a query, would assemble a slightly different set of genes. If we take first line as a starting set than we shall add to this set all the genes that would have been selected if each member of the set were used as a query.

Superfamily selection scheme works as follows: it takes one line, for each gene in the line it searches for the occurrence of this gene in all other lines. The content from all intersecting lines is combined in one superfamily. All intersected lines are removed from the file, and assemblage starts from the next available line.

EXAMPLE: 13 gamma proteobacteria

For the case of 13 different species we run the script with the requirement that at least 8 different taxa should be present:

```
> perl parse_superfamilies.pl 8
```

As a result file `parsed/all_vs_all.fam` has 1205 lines, i.e. 1205 superfamilies.

IV. Reconstruction of superfamilies trees

At this point list of all superfamilies is stored in the file `parsed/all_vs_all.fam`. Each line of the file represents one superfamily.

For each superfamily we do alignment and reconstruct a phylogenetic tree.

1. Prepare files containing all gene sequences for a given superfamily:

```
> perl prepare_fa.pl <FILE-WITH-SUPERFAMILIES>
```

Here `<FILE-WITH-SUPERFAMILIES> is all_vs_all.fam`

```
Result: fa/fam_##.fa
```

EXAMPLE: 13 gamma proteobacteria

```
> perl parse_superfamilies.pl parsed/all_vs_all.fam
```

2. Alignment of superfamilies:

```
> perl do_clustalw_aln.pl
```

```
Result: dist/fam_##.aln
```

3. Trees reconstruction using distance method with Kimura correction:

Note! Distance method here used for speed, it can be replaced with any other method of tree reconstruction and any other program (phyml, tree-puzzle, paup, etc.). However, for the purpose of clustering, when we are not looking at the evolutionary history inside each family, the results obtained with the distance method are not different from those obtained with the maximum likelihood method.

```
> perl do_clustalw_dist_kimura.pl
```

```
 Result: dist/fam_##.ph
```

4. Prepare trees to use further with BranchClust:

```
> perl prepare_trees.pl
```

```
 Result: trees/fam_##.tre
```

V. Selection of orthologous families with BranchClust

1. BranchClust method:

```
> perl branchclust_all.pl <MANY>
```

```
Results: clusters/clusters_##.out
         clusters/family_##.list
Additional: clusters_##.log
```

Program branchclust_all.pl is a wrapper that applies BranchClust algorithm to each superfamily tree. Requires branchclust.pl located in the same directory. Requires taxa identification file, `gi_numbers.out`.

Parameter <MANY> is a number less or equal to the number of different taxa in question. See description of the BranchClust algorithm for the choice of this parameter. In general it is recommended to use around 50-80% of the number of different taxa.

Each superfamily tree is analyzed and divided into clusters. Each cluster contains a potential family of orthologs together with paralogs. Here it is the typical BranchClust output for superfamily consisting of one cluster.

```
------------ CLUSTER -----------
30995452  16273249  15602639  15602307  15598924  21243223  21230752  15601922  15601924  16123078
22125245 22125244 16766857 26250079 16082793 16767596 16764401 16082788

------------ FAMILY ------------
26250079 30995452 15602639 15598924 16766857 21230752 21243223 16123078 22125245
INCOMPLETE: 9

>>>>> IN-PARALOGS -----------
      16273249 15602307 15601922 15601924 22125244

<<<<< OUT-OF_CLUSTER PARALOGS -----------
      16082793 16767596 16764401 16082788
```

First the cluster itself is reported then the potential family of orthologs with in- and out-of-cluster paralogs. The following relation is satisfied: `CLUSTER = FAMILY + IN_PARALOGS + OUT-OF-CLUSTER PARALOGS`. Superfamily could be composed of many clusters, in this case all the clusters are printed in order into one file.

Each result file `Clusters_##.out` corresponds to one superfamily.

In addition to clusters, the list of all families assembled from one superfamily is printed in a separate file `family_##.list`. Here it is the typical example of family list:

```
26248867 15601281 15597585 16764354 15600941 21233415 21244798 15838496 16120414 22124883
INCOMPLETE: 10
27904517  26250482  16272430  15603355  15600751  16767153  15642761  32491162  21230753  21244378
15837749 16124233
22128007
COMPLETE: 13
```

Each taxon is represented only once per a family. All complete, i.e. containing all taxa, and incomplete, i.e. missing some taxa, families are reported.

You can apply BranchClust algorithm to only one tree. It is done with the program branchclust.pl:

```
> perl branchclust.pl <TREE> <MANY>
```

```
Results: clusters.out
         family.list
Additional: clusters.log
```

EXAMPLE: 13 gamma proteobacteria

For the case of 13 different species we run the program with parameter MANY = 8

```
> perl branchclust_all.pl 8
```

If you want to try different values of parameter MANY for one superfamily then you use branchclust.pl program and pass the superfamily tree as a first parameter and MANY as the second:

```
> perl branchclust.pl trees/fam_16.tre 5
```

2. Make a list of all families:

```
> perl make_fam_list.pl <NU_OF_ALL_TAXA> <NU_OF_INCOMPLETE_TAXA>
```

```
<NU_OF_ALL_TAXA> - number of different taxa
<NU_OF_INCOMPLETE_TAXA>  -  even  incomplete  families  with  missing  genomes
will be included.

Result: families.list
```

EXAMPLE: 13 gamma proteobacteria

If we would like to print out only complete families where all species are present then the command line is:

```
> perl make_fam_list.pl 13 13
```

If we would like to print out also incomplete families with at least 7 taxa being present then the command line is:

```
> perl make_fam_list.pl 13 7
```

3. Analysis of clusters and selected families:

Prints out how many gene families were assembled in general, and reports the numbers for complete and incomplete collections:

```
> perl summary.pl
```

```
Result: summary.out
```

EXAMPLE: 13 gamma proteobacteria

```
> perl summary.pl
> more summary.out

complete: 366
incomplete: 2592
total: 2958
------ details -------
incomplete 12: 242
incomplete 11: 405
incomplete 10: 318
incomplete 9: 339
incomplete 8: 395
incomplete 7: 117
incomplete 6: 55
incomplete 5: 62
incomplete 4: 103
incomplete 3: 118
incomplete 2: 152
incomplete 1: 286
```

For each cluster you can print gene names taken from original fasta-files:

```
> perl names_for_cluster_all.pl
```

```
Results: clusters/clusters_##.out.names
```

You can print gene names for the individual file:

```
> perl names_for_cluster.pl <CLUSTER-FILE>
```

```
Results: <CLUSTER-FILE>.names
```

EXAMPLE: 13 gamma proteobacteria

```
> perl names_for_cluster.pl clusters/clusters_6.out
> more clusters/clusters_6.out.names
...
------------ FAMILY ------------
>gi|27904576|ref|NP_777702.1| flagellum-specific ATP synthase [Buchnera aphidicola str. Bp
(Baizongia pistaciae)]
>gi|26248210|ref|NP_754250.1| Flagellum-specific ATP synthase [Escherichia coli CFT073]
>gi|15596301|ref|NP_249795.1| flagellum-specific ATP synthase FliI [Pseudomonas aeruginosa
PAO1]
>gi|16765310|ref|NP_460925.1| flagellum-specific ATP synthase [Salmonella typhimurium LT2]
>gi|15642129|ref|NP_231761.1| flagellum-specific ATP synthase FliI [Vibrio cholerae O1
biovar eltor str. N16961]
>gi|32490806|ref|NP_871060.1| hypothetical protein WGLp057 [Wigglesworthia glossinidia
endosymbiont of Glossina brevipalpis]
>gi|21231371|ref|NP_637288.1| flagellar protein [Xanthomonas campestris pv. campestris str.
ATCC 33913]
>gi|21242695|ref|NP_642277.1| flagellar protein [Xanthomonas axonopodis pv. citri str. 306]
>gi|16122080|ref|NP_405393.1| flagellum-specific ATP synthase [Yersinia pestis CO92]
>gi|22126362|ref|NP_669785.1| flagellum-specific ATP synthase [Yersinia pestis KIM]

INCOMPLETE: 10
>>>>> IN-PARALOGS -----------
<<<<< OUT-OF_CLUSTER PARALOGS -----------
>gi|22127336|ref|NP_670759.1| flagellum-specific ATP synthase [Yersinia pestis KIM]
>gi|16121036|ref|NP_404349.1|  putative  flagellum-specific  ATP  synthase  [Yersinia  pestis
CO92]
>gi|22124440|ref|NP_667863.1| flagellum-specific ATP synthase [Yersinia pestis KIM]
>gi|16120605|ref|NP_403918.1|  putative  type  III  secretion  system  ATP  synthase  [Yersinia
pestis CO92]
>gi|16764765|ref|NP_460380.1| needle complex ATPase [Salmonella typhimurium LT2]
>gi|21241185|ref|NP_640767.1| HrcN protein [Xanthomonas axonopodis pv. citri str. 306]
>gi|21230693|ref|NP_636610.1|  HrpB6  protein  [Xanthomonas  campestris  pv.  campestris  str.
ATCC 33913]
>gi|16082728|ref|NP_395174.1| needle complex secretion ATPase [Yersinia pestis CO92]
>gi|15596894|ref|NP_250388.1|  ATP  synthase  in  type  III  secretion  system  [Pseudomonas
aeruginosa PAO1]
>gi|16766200|ref|NP_461815.1| needle complex secretion ATPase [Salmonella typhimurium LT2]

…
```

Detailed summary prints out the table where each row contains information about the number of genes in the cluster, number of genes in the family, number of paralogs and the name associated with the family extracted from an arbitrarily taken gene:

```
> perl detailed_summary.pl
```

Result: `detailed_summary.out`

EXAMPLE: 13 gamma proteobacteria

```
> perl detailed_summary.pl
> more detailed_summary.out

superfamily_##  family_##       nu_of_genes_in_the_family       nu_of_paralogs  family_name
1       1       13      0       glucose-inhibited division protein A
10      1       13      0       DNA polymerase III, beta-subunit
100     1       11      0       50S ribosomal protein L35
1000    1       10      2       Hypothetical protein yhhQ
1001    1       10      7       FixC protein
1002    1       8       0       Hypothetical protein yhiQ
1003    1       11      0       Hypothetical protein yhiR
1004    1       7       1       Probable cytochrome C peroxidase
1005    1       10      1       DNA-3-methyladenine glycosylase I
1006    1       8       4       Hypothetical acetyltransferase yiaC
1007    1       7       4       Putative hexulose-6-phosphate isomerase
1008    1       11      1       tRNA (Guanosine-2'-O-)-methyltransferase
1009    1       12      0       Glycerol-3-phosphate dehydrogenase
101     1       13      0       RpL20
1010    1       11      0       Protein-export protein secB
1011    1       11      20      Lipopolysaccharide core biosynthesis glycosyl transferase
1012    1       12      0       3-deoxy-D-manno-octulosonic-acid transferase
1013    1       12      0       Phosphopantetheine adenylyltransferase
1014    1       12      0       DNA/pantothenate metabolism flavoprotein
1015    1       11      10      Putative conserved protein
1016    1       12      0       Protein yicC
1017    1       12      0       DNA-directed RNA polymerase omega chain
1018    1       7       0       Sodium/glutamate symport carrier protein
1019    1       8       2       Putative symporter yjcG
1019    2       8       7       Sodium/pantothenate symporter
1019    3       8       2       Putative symporter yidK
…
…
```

VI. Results visualization with TreeDyn

Graphic representation of a clustered tree can be automated using the TreeDyn program. One can color different clusters in different colors and label leaves in a superfamily tree according to their belonging either to a family or to in- or out-of-cluster paralogs. Additionally, one can include gene annotation and species names.

TreeDyn requires an annotation file for a given tree and a script with TreeDyn operations. You must first obtain names for the genes in a cluster (see `names_for_cluster.pl` above).

Generation of TreeDyn annotation file:

```
> perl generate_tlf.pl <CLUSTER_FILE.names>
```

```
Result: labelfile.tlf
```

Example of script with TreeDyn operations is given in the file `commands.tds`.

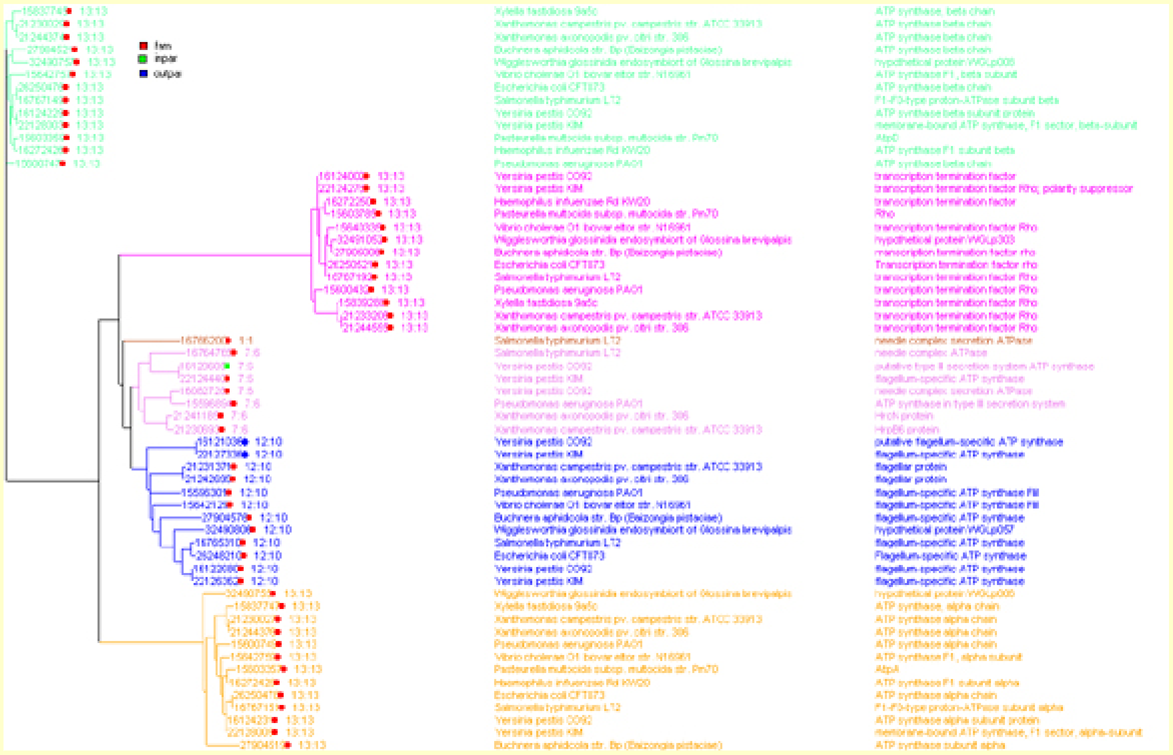
In TreeDyn program:
1. `File/Open tree(s)…` - Open tree-file (from `trees/fam_##.tre`)
2. `File/Load annotations …` Load file `labelfile.tlf`
3. Menu `File/Load script…` Load script with operations `commands.tds`.

EXAMPLE: 13 gamma proteobacteria

```
> perl generate_tlf.pl clusters/clusters_6.out.names
```

In TreeDyn open tree-file `fam_6.tre`, load annotation file `labelfile.tlf`, open script `commands.tds`. You can save the created image as *.png.

Result of BranchClust selection for the `superfamily #6`.



fam
inpar
outpar