Genomics - Walkthrough

Notebook:	Thesis Methods		
Created:	15/09/2020 16:11	**Updated:**	15/09/2020 17:36
Author:	Jess Friedersdorff		

Genomics - Phylogenetic Tree Using Set of 40 Conserved Genes
Building a BLAST database:

- First, build a BLAST database using a file containing all of the template genes for each of the 40 cogs.
    - (There are over 1000 template genes, each from a different organism or species which these 40 genes have been found in).
- To do this, create a FASTA file and use DIAMOND (DIAMOND V0.7.9.):

```
for f in *.fa; do sed -i "s/>/>$f\s/" "$f"; done

cat *.fa > ALLCOG.faa

diamond makedb --in ALLCOG.faa -d ALLCOG_database

for i in prokka_*; do cd $i; for j in *.faa; do sed "s/>/>$j./g" $j; done; cd
../; done > faafiles.faa
```

- The database was then queried using the .faa file of (all of the) Hungate species' genes using DIAMOND with the following script:

```
diamond blastp -d path/to/database.dmnd -q path/to/faafiles.faa -a path/to/output
diamond view -a path/to/input -o pat/to/output.tab -f tab
```

Nearly all of the 494 genomes have all 40 of The 40 Genes. The fewest that a bacteria had was 31:

```
more Hungatererun_vs_40genes_results.tab | awk 'BEGIN{prev=""};{if($1 != PREV){
print $0; PREV=$1;}}' > tophits_cog.txt
```

- Since it is always sorted by Bitscore, the top hit for each gene is the best. This does mean that a single cog might occur more than once in more than one gene per genome and doesn't account for this.

```
cat tophits_cog.txt | sed 's/sp\./sp/g' | awk -f ./awk_script | sort >
cogsforeachbacteria.txt
```

Where the awk script was:

```
#Script to give to awk.
#finds the highest bit score (value in the 12th column) for every unique
combination of bacteria and cog (column 1 and column 2)

{
sub("\\..*", "", $1);
sub("\\..*", "", $2);
if ($12 > vals[$1"."$2]) {
vals[$1"."$2] = $12;
}
}
```

```
END {
for (key in vals) {
print key "\t" vals[key];
}
}
```

- The result was a file, with lines consisting of the name of each bacteria, the name of the cog and the bit score that cog best hit at. Then, the amount of cogs for each bacteria was counted:

```
cat cogsforeachbacteria.txt | cut -f1 -d'.' | uniq -c >
count_cogsforeachbacteria.txt
```

    - Which lead to the findings that nearly all of the 494 have all 40 of The 40 Genes. The fewest that a bacteria had was 31.

Extracting Data to Use Further:
40 individual FASTA files, one for each cog, containing the sequences in the Hungate genomes was made.

- Ordered the query blast results first by cog name, then organism name, then bitscore:

```
sed 's/\.fa[as]/ /g' Hungatererun_vs_40genes_results.tab | sort -k3,3 -k1,1 -
k14,14rn > Hungatererun_vs_40genes_results.sorted.tab
```

- Then, using awk, find the first instance of the cog in the genome, keeping the gene name. Only those with bitscores above 60 are kept. (Some cogs only hit once per genome and some of these had low bitscores. Setting it to 60 removes those. Some missing data like this is not a problem when building trees)

```
cat Hungatererun_vs_40genes_results.sorted.tab | awk 'PREV{genome=""; cog=""};
{if($1 != genome || $3 != cog) { genome=$1; cog=$3; if($14 > 60) { print $0}}}' >
Hungatererun_vs_40genes_results.matches.tab
```

- So that the .faa file of Hungate genes can be searched, the file needs to be changed from a two line FASTA to a single line FASTA (name \t sequence \n name \t sequence etc).

```
$ cat faafiles.faa | cut -d' ' -f1 | sed '/>/s/$/\t/g' | sed 's/>/#>/g' | tr -d
'\n' | tr '#' '\n' | sed '1d' | tr -d '>' > editedfaafiles.faa
```

- Using the file with the best matches of 40 genes to Hungate genomes, search the Hungate file of genes, and extract the name and sequence, and output this to to 40 different files, one for each Cog.

```
$ cat Hungatererun_vs_40genes_results.matches.tab | awk 'BEGIN{ while(getline <
"editedfaafiles.faa"){ split($0, a, "\t"); array[a[1]]=a[2]; }};{ printf
">"$1$2"\n"array[$1".faa"$2]"\n" >> $3".faa" }'
```

Alignment of 40 COGs using MUSCLE:
MUSCLE v3.8.31 (Edgar, R.C. Nucleic Acids Res 32(5), 1792-97)
Example command:

```
$ muscle -in COG0012.faa.1 -out COG0012.faa.1.aligned
```

Concatentating Sequences Together using Catsequences
Now all of the sequences need concatenating together, to create one FASTA alignement file, where each entry in the file is one long sequences of the aligned 40 genes pasted together into one long sequence.
To do this, a programme called Catsequences was used, available at Creevey GitHub.

- A list of files was created in the directory COG.faa.aligned_files:

```
ls *COG*.faa_aligned > listoffiles.txt
catsequences.exe listoffiles.txt
cat allseqs.fas | tr '?' '-' > allseqs_Hungate_Cog.fas
```

This file was then copied into the main directory (40genes) and given to RAxML.

Tree Building with RAxML:
RAxML version 8.2.9
Tree was built using this command:

```
raxmlHPC-PTHREADS-SSE3 -f a -N 100 -x 6543 -T 4 -m PROTGAMMALG -p 12345 -s
allseqs_Hungate_Cog.fas -n RAxML_COG2_bootstraps
```

Resulting tree could be seen here: https://itol.embl.de/tree/144124105664448815088838827 (as of 15th September 2020)

Making Count Tables:

Requires a file containing a list of things to count, the file(s) where these things should be counted in, then querying whether the thing to be searched is present, so that if it is not, this can also be recorded, and if it is, then it can be counted. for each file, these are outputted to a new file that contains just the two columns (or even one column with counts) then are pasted together to make a new table.

For example, for EGN gene family counts (where HAP_NAP_SAP.list contains the species of interest which are the things to count and the .faa are gene family files containing the things that should be counted):

```
for j in *.faa; do
    for i in $(cat HAP_NAP_SAP.list); do
        count=$(if grep -q ${i} ${j};
                then grep -c ${i} ${j};
                else echo "x";
            fi);
        echo -e ${j}"\t"${count} >> ${i}.col;
    done;
done


for i in *.col; do
    echo ${i} > ${i}.edit;
    cat ${i} >> ${i}.edit;
done


paste *.col.edit > HAP_NAP_SAP_EGN_counts.table
```

bactCOGs and functional counts in genomes

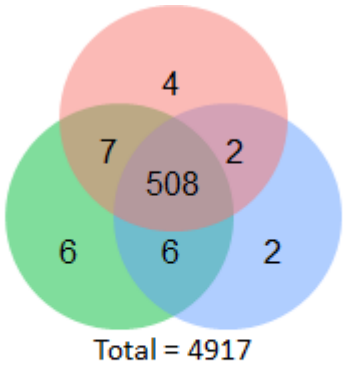
Use the @2 (bacterial functions) as a high up hierarchy of bacterial groups:

First make a mapping file, for all the genes with the "@2" function:

```
for i in *.annotations; do awk 'BEGIN{FS=OFS="\t"}{split($19, a, ","); for (i in
a) {if(a[i] ~ /@2$/) {print $1, a[i]} } } ' ${i}; done > gene_to_bactOG_map.list

output: (one line per COG, not per gene!)
ECADDMPC_03180  COG1690@2
ECADDMPC_03181  COG2304@2
ECADDMPC_03182  COG2819@2
ECADDMPC_03183  COG4870@2
ECADDMPC_03183  COG5263@2
ECADDMPC_03184  COG1578@2
ECADDMPC_03185  COG4633@2

Then make one per sample:
for i in *.annotations; do awk 'BEGIN{FS=OFS="\t"}{split($19, a, ","); for (i in
a) {if(a[i] ~ /@2$/) {print $1, a[i]} } } ' ${i} > ${i}_bactOGonly; done

also make a list of all the bactOGs that have occured in the annotations:
for i in *.annotations; do awk 'BEGIN{FS=OFS="\t"}{split($19, a, ","); for (i in
a) {if(a[i] ~ /@2$/) {print $1, a[i]} } } ' ${i}; done | cut -f2 | sort | uniq >
list_of_bactOGs.list

wc -l list_of_bactOGs.list
4916 list_of_bactOGs.list
There are 4916 bacterial OGs. It seems that an annotation can't have an X@2 and a
COGX@2. but can have multiple COGX@2.

Then for each of those bactOGs that have occurred in any sample, count how many
times they occur in each sample:
for i in *bactOGonly; do for j in $(cat list_of_bactOGs.list); do count=$( grep -
wc ${j} ${i}); echo -e ${j} "\t" ${count} ; done > ${i}.col; done

paste *.col > bactOG_counts.tsv
```

Counts of bactCOG families shared between the ammonia phenotype groups, where all three species must have a representative.

shared/unique OGs	HAP	SAP	NAP	ALL
HAP	4	2	7	**508** **Total = 4917**
SAP	2	2	6	
NAP	7	6	6	



Number of Bacterial orthologous genes (OGs) shared between and unique to HAPs, NAPs, SAPs

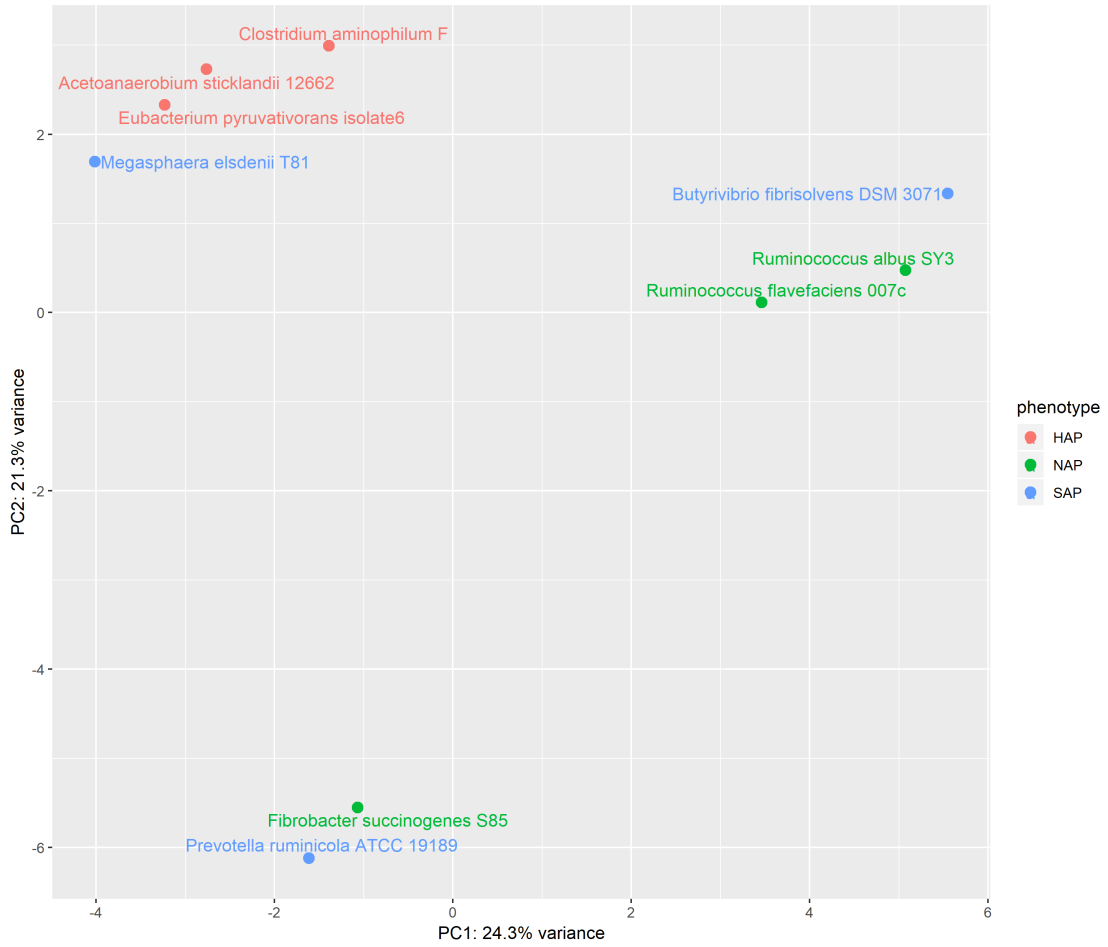
4 COGs for HAPs

COG0384@2 - isomerase, unknown function
COG1292@2 - Cell wall/membrane/envelope biogenesis; Belongs to the BCCT transporter (TC 2.A.15) family
COG2202@2 - Signal transduction mechanisms; Pas domain
COG5505@2

2 for HAPs SAPs

COG1063@2 - Amino acid transport and metabolism; alcohol dehydrogenase
COG4122@2 - Amino acid transport and metabolism; O-methyltransferase activity

R:
Use the R script genomes_bact_OGs_pcas.R



Loadings:

7 components needed to describe 94.285% of variance.
COG0395 influences PC1 most - inorganic ion transport and metabolism; glycerophosphodiester transmembrane transport
COG0823 influences PC2 most - Intracellular trafficking, secretion, and vesicular transport; Involved in the tonB-independent uptake of proteins

---

Making proportional stacked graphs for COG functional identifiers in genomes

First make a count table for the numbers of genes in each of the categories for each of the genomes: (COG Functional Category)

```
INFORMATION STORAGE AND PROCESSING
[J] Translation, ribosomal structure and biogenesis
[A] RNA processing and modification
[K] Transcription
[L] Replication, recombination and repair
[B] Chromatin structure and dynamics


CELLULAR PROCESSES AND SIGNALING
[D] Cell cycle control, cell division, chromosome partitioning
[Y] Nuclear structure
[V] Defense mechanisms
[T] Signal transduction mechanisms
[M] Cell wall/membrane/envelope biogenesis
[N] Cell motility
[Z] Cytoskeleton
[W] Extracellular structures
```

```
[U] Intracellular trafficking, secretion, and vesicular transport
[O] Posttranslational modification, protein turnover, chaperones


METABOLISM
[C] Energy production and conversion
[G] Carbohydrate transport and metabolism
[E] Amino acid transport and metabolism
[F] Nucleotide transport and metabolism
[H] Coenzyme transport and metabolism
[I] Lipid transport and metabolism
[P] Inorganic ion transport and metabolism
[Q] Secondary metabolites biosynthesis, transport and catabolism


POORLY CHARACTERIZED
[R] General function prediction only
[S] Function unknown
```

obtained from: ftp://ftp.ncbi.nih.gov/pub/wolf/COGs/COG0303/fun.txt (accessed 06/01/2020)

```
for i in $(cat COG_functional_identifiers.list | cut -f1); do for j in
*.annotations; do name=$(echo ${j} | cut -d "." -f1); count=$(cut -f21 ${j} |
grep -c ${i}); echo -e ${i}"\t"${count} >> ${name}_func.col; done; done
for i in *.col; do echo -e "\t" ${i} > ${i}.edit; cat ${i} >> ${i}.edit; done
paste *.edit > COG_functional_identifiers_counts.table
```

Use the insert -> graph -> percentage stacked graph -> switch rows/columns to get a stacked graph in Excel.

Transporters - Amino acids and peptides.

Table of transporter codes:

1.B.83	The Alpha-Aminoxy Acid Channel (AAAC) Family	AAAC
2.A.118	The Basic Amino Acid Antiporter (ArcD) Family	ArcD
2.A.3	The Amino Acid-Polyamine-Organocation (APC) Family	APC
2.A.95	The 6 TMS Neutral Amino Acid Transporter (NAAT) Family	NAAT
3.A.1.3	The Polar Amino Acid Uptake Transporter (PAAT) Family	PAAT
3.A.1.4	The Hydrophobic Amino Acid Uptake Transporter (HAAT) Family	HAAT
2.A.120	The Putative Amino Acid Permease (PAAP) Family	PAAP
2.A.42	The Hydroxy/Aromatic Amino Acid Permease (HAAAP) Family	HAAAP
2.A.18	The Amino Acid/Auxin Permease (AAAP) Family	AAAP
2.A.23	The Dicarboxylate/Amino Acid:Cation (Na^+ or H^+) Symporter (DAACS) Family	DAACS
2.A.26	The Branched Chain Amino Acid:Cation Symporter (LIVCS)	LIVCS

	Family	
2.A.21.2	The Solute:Sodium Symporter (SSS) Family	SSS
2.A.25	The Alanine or Glycine:Cation Symporter (AGCS) Family	AGCS
2.A.78	The Branched Chain Amino Acid Exporter (LIV-E) Family	LIV-E
2.A.114	The Putative Peptide Transporter Carbon Starvation CstA (CstA) Family	CstA
2.A.67	The Oligopeptide Transporter (OPT) Family	OPT
2.A.17	The Proton-dependent Oligopeptide Transporter (POT/PTR) Family	POT/PTR
1.A.11	Ammonium channel	Amt

Get all the tcdb fasta files:
http://www.tcdb.org/public/tcdb downloaded 14/01/2020,

- TCDB FastA Sequences *(Last modified: January 14 2020)*

Made a blast database:

```
makeblastdb -in tcdb.fasta -out tcdb -dbtype prot
```

Make submission file and blast the .faa files of the HAPs, NAPs and SAPs against the database:

```
subheader.sh blast_tcdb 25G > blast_tcdb.sub
for i in *.faa; do echo "blastp -query "${i}" -db tcdb -out "${i}"_tcdb -outfmt
6"; done >> blast_tcdb.sub
```

Get tophits of results:

```
for i in *tcdb; do awk 'BEGIN {prev=""; score=""} {if ($1 != prev && $12 >= 60)
{print $0; prev=$1}}' ${i} > ${i}_tophits; done #Remove any hits that are under
60 bitscore
```

make a file of just the amino acid transporters codes.

```
for i in *_tophits; do grep -f aatransporters.list ${i} >
${i}_aatransportersonly; done
```

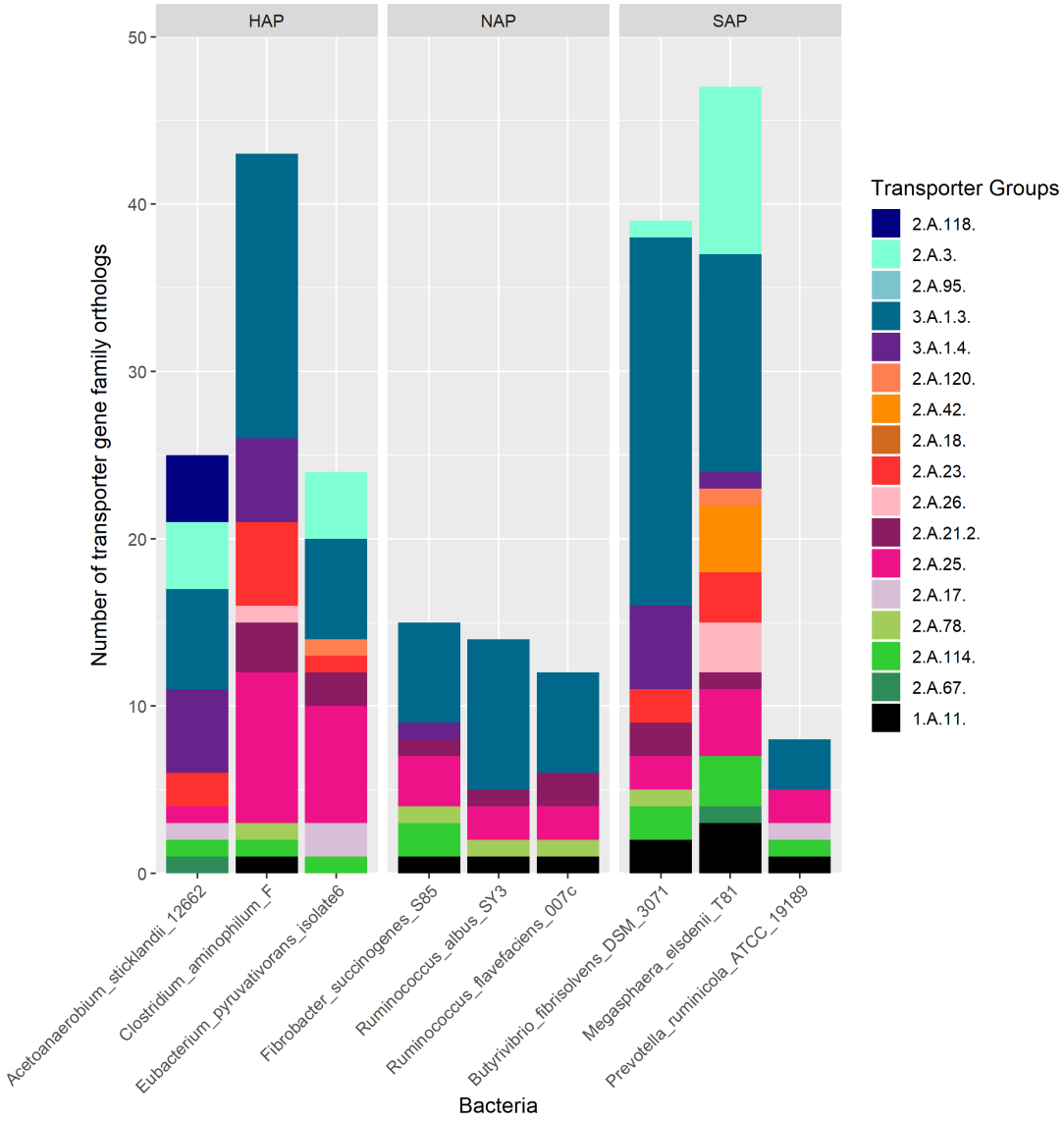
Count the occurrences of each of the transporters:

```
for i in *_tophits; do name=$(echo ${i} | cut -d "." -f1); echo -e
"transportercode" "\t" ${name} > ${name}_aatransporter.col; for j in $(cat
aatransporters.list); do count=$(grep -c ${j} ${i}); echo -e ${
j}"\t" ${count} >> ${name}_aatransporter.col; done ; done
```

Paste columns and make table:

```
paste *col > aa_transporter_counts.tsv
```

Then run the R script genomes_aatransporters.R and make the graph:


HAP
NAP
SAP
Number of transporter gene family orthologs
0
10
20
30
40
50
Acetoanaerobium_sticklandii_12662
Clostridium_aminophilum_F
Eubacterium_pyruvativorans_isolate6
Fibrobacter_succinogenes_S85
Ruminococcus_albus_SY3
Ruminococcus_flavefaciens_007c
Butyrivibrio_fibrisolvens_DSM_3071
Megasphaera_elsdenii_T81
Prevotella_ruminicola_ATCC_19189
Bacteria
Transporter Groups
2.A.118.
2.A.3.
2.A.95.
3.A.1.3.
3.A.1.4.
2.A.120.
2.A.42.
2.A.18.
2.A.23.
2.A.26.
2.A.21.2.
2.A.25.
2.A.17.
2.A.78.
2.A.114.
2.A.67.
1.A.11.