Infant Gut Microbiome Project Notebook

Elizabeth McDaniel

2022-04-04

Contents

Transposons in the Infant Gut Microbiome

I hypothesized that transposons/transposases would be highly prevalent mobile elements in the infant gut microbiome, and could potentially facilitate inter-species horizontal gene transfer. I first performed an initial inspection of the contigs to look at the taxonomical makeup of the community and any possible contamination. I then performed functional annotation with the KofamKOALA and mobileOG databases to both annotate all proteins that could be annotated with the KEGG database and all mobile elements using the mobileOG curated database. To then look for putative transposase-mediated signatures of horizontal gene transfer, I identified transposases and performed BLASTP searches to find gene neighborhoods of 5-10 genes surrounding these transposases with homology to gene neighborhoods in other species.

Initial Contigs Inspection

First I inspected the contigs to look at the data I was dealing with:

```
head 0118A1023.contigs.fa
```

```
>qb307082014_I_scaffold_1012 id=7301749 bin="0118A1023_UNK"
CCCGGCGTACGGGGCTGGGATTTGACGTCAACCGATTGCATCTTATCTTGGCTGTATTGG
AAAAACGCCTGCGCCTGAACTTCGGTCAGGTCGACATTTATGCCAAGGTCGGCGGCGGCA
TGAAGATTCAGGAGCCGGGCATGGACCTCGCGCTCGTGGCGGCGATGTTGTCGTCCTTCT
ACGACGTGCCGCTTCCCGAGCGGGCCGTGCTGTGGGGCGAAGTGGACCTCAACGGTCAGA
TCCGCCCCGTGGCCGCGCACGATATCCGGCTTTCGCAGGCGCGCAGGCTTGGCTACAAAC
CGATCCTTTTTCCTTCGCAGGGCGAGGGCGACGGAATCGCCACGGTCGTGGAGTTGCAGG
ACAGGCTGTTCCGCCGCAAGAACTGACGGCGGAAGCGGAAAGGGCCTCAAGTCGGGCGGG
CGCGGATCTCCACGGGGCCGGTTTTGGCGTGTGGGGAAAAAAATGCATGACGCTCCCCTGT
TGCTCTTGCGCGGCCCTTGCGAACGCGTTATCAGGAATTATTCTTTATTCAAAAGGACGA
```

Since it looked like this was a file that had already been assembled into genome bins, I wanted to know how many genomes were within this assembly and how many scaffolds each genome had:

```
grep '>' 0118A1023.contigs.fa | awk -F " " '{print $3}' | sort | uniq -c
```

Which produced:

```
 157 bin="0118A1023_Bacteroides_vulgatus-like_42_520"
  23 bin="0118A1023_Citrobacter_koseri_53_430"
 922 bin="0118A1023_Clostridium_difficile_28_5"
  93 bin="0118A1023_Enterococcus_faecalis_36_53"
   5 bin="0118A1023_Enterococcus_faecalis_phage"
   1 bin="0118A1023_Klebsiella_pneumoniae_plasmid"
   7 bin="0118A1023_Proteobacteria_phage"
 104 bin="0118A1023_UNK"
```

It appears that there are 8 genomes in total, 4 of which are bacteria, 2 are phage, 1 a plasmid, and 1 unknown genome. To confirm this and possibly get a better idea of what the "unknown" genome is, I profiled the contigs with Anvi'o.

Anvi'o doesn't like long contig names, so I first created a plain file connecting each scaffold name to each bin name:

`grep '>' 0118A1023.contigs.fa | awk -F " " '{print $1"\t"$3}' > infant-gut.stb`, which looks like:

```
>qb307082014_I_scaffold_1012    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1021    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1039    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1068    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1069    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1074    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1094    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1098    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1104    bin="0118A1023_UNK"
>qb307082014_I_scaffold_1110    bin="0118A1023_UNK"
```

Then I shortened the contig names with `sed 's/\s.*$//' 0118A1023.contigs.fa > infant-gut-contigs.fa` so now the contig names look like:

```
>qb307082014_I_scaffold_1012
>qb307082014_I_scaffold_1021
>qb307082014_I_scaffold_1039
>qb307082014_I_scaffold_1068
>qb307082014_I_scaffold_1069
>qb307082014_I_scaffold_1074
>qb307082014_I_scaffold_1094
>qb307082014_I_scaffold_1098
>qb307082014_I_scaffold_1104
>qb307082014_I_scaffold_1110
```

I then generated a contigs database in Anvi'o with `anvi-gen-contigs-database -f infant-gut-contigs.fa -o contigs.db -n INFANTGUT -T 4`. This will identify ORFs in contigs with prodigal, which produced:

```
Result ......................................: Prodigal (v2.6.3) has identified 17152
genes.
```

I then wanted to know the distribution of single copy core genes among these genomes, and what their taxonomy is. First I ran `anvi-run-hmms -c contigs.db --num-threads 4` which identifies sets of HMMs in the pre-made databases for `Ribosomal_RNA_16S, Ribosomal_RNA_5S, Ribosomal_RNA_28S, Protista_83, Ribosomal_RNA_12S, Archaea_76, Bacteria_71, Ribosomal_RNA_18S, Ribosomal_RNA_23S`. This pipeline can also use HMMs made from other sources, but the `Bacteria_71` and ribosomal sets will suffice for inspecting the contigs.

To then assess the taxonomy of these single copy core genes in the metagenome, I first ran `anvi-run-scg-taxonomy -c contigs.db --num-threads 4 --min-percent-identity 90 --all-hits-output-file` `scg-taxonomy.txt`. This will return all taxonomy classifications to the single copy core genes with at least 90% identity based on the Genome Taxonomy Database (GTDB). This took less than 30 seconds with 12 threads. This output is in `results/scg-taxonomy.txt` which gives any taxonomical result matching the given single copy core gene with above 90% identity.

I then used this information to assess the taxonomical composition of the entire metagenome based on the classification of a single copy core gene using: `anvi-estimate-scg-taxonomy -c contigs.db -T 4 --metagenome-mode -o infant-gut-tax.txt` which will use one of the single copy core genes (ribosomal L2 in this case), which the output looks like (the file is also viewable in `results/infant-gut-tax.txt`):

```
scg_name    percent_identity    t_domain    t_phylum    t_class t_order t_family    t_genus t_species
Ribosomal_L2_3113   99.6    Bacteria    Firmicutes  Clostridia  Peptostreptococcales    Peptostreptococ
Ribosomal_L2_14412  98.5    Bacteria    Proteobacteria  Gammaproteobacteria Enterobacterales    Enteroba
Ribosomal_L2_792    98.9    Bacteria    Bacteroidota    Bacteroidia Bacteroidales   Bacteroidaceae  Phoc
Ribosomal_L2_8091   99.6    Bacteria    Firmicutes  Bacilli Lactobacillales Enterococcaceae Enterococcus
Ribosomal_L2_5822   99.1    Bacteria    Desulfobacterota    Desulfovibrionia    Desulfovibrionales  Desu
```

From this check, there are 5 Ribosomal L2 single copy genes in the metagenome, and these are the classifications. To ensure that these classifications match up with the classifications given for the bins in the initial scaffolds file, we can export the gene calls and HMM hits from Anvi'o with the scripts:

`anvi-export-gene-calls -c contigs.db --gene-caller prodigal -o gene_calls.txt` `anvi-script-get-hmm-hits-p` `-c contigs.db -o hmm-hits.txt --hmm-source Bacteria_71`

I then used these files to import into R for some basic stats and comparing the taxonomy of the Ribosomal L2 gene to the given taxonomy in the contigs file.

I imported the gene calls and corresponding scaffolds to bins into R, and joined these tables to create a table of all genes for all corresponding bins:

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
# import gene calls, scaffold names, and corresponding scaffolds to bins
gene_calls <- read.table("results/gene_calls.txt", header=TRUE, sep="\t") %>%
  select(gene_callers_id, contig)

scaffold_to_bin <- read.table("raw_data/infant-gut.stb", col.names = c("contig", "bin"), sep="\t") %>%
  mutate(contig = gsub(">", "", contig))

# full table of gene accessions for each bin
genes_table <- left_join(gene_calls, scaffold_to_bin)
```

```
## Joining, by = "contig"
```

```r
head(genes_table)
```

```
##   gene_callers_id                      contig               bin
## 1               0 qb307082014_I_scaffold_1012 bin=0118A1023_UNK
## 2               1 qb307082014_I_scaffold_1012 bin=0118A1023_UNK
## 3                2 qb307082014_I_scaffold_1012 bin=0118A1023_UNK
## 4                3 qb307082014_I_scaffold_1117 bin=0118A1023_UNK
## 5                4 qb307082014_I_scaffold_1168 bin=0118A1023_UNK
## 6                5 qb307082014_I_scaffold_1168 bin=0118A1023_UNK
```

I then imported all HMM hits for the Bacteria_71 set to both confirm the classifications for the Ribosomal L2 gene and do some statistics on how many single copy core genes were found in the contigs for each bin:

```
# HMM hits for Bacteria_71 collection to look at Ribosomal L2 hits and aggregating by genome to assess
hmm_hits <- read.table("results/hmm-hits.txt", header=TRUE, sep="\t") %>%
  select(gene_callers_id, source, gene_name, gene_hmm_id)

# join the genes table with the HMM hits to correspond with bin information
hmm_table <- left_join(hmm_hits, genes_table)
```

```
## Joining, by = "gene_callers_id"
```

```
hmm_table %>%
  select(gene_name) %>%
  unique() %>%
  count()
```

```
##    n
## 1 71
```

```
hmm_table %>%
  group_by(bin) %>%
  unique() %>%
  count()
```

```
## # A tibble: 5 x 2
## # Groups:   bin [5]
##   bin                                                 n
##   <chr>                                           <int>
## 1 bin=0118A1023_Bacteroides_vulgatus-like_42_520    73
## 2 bin=0118A1023_Citrobacter_koseri_53_430           71
## 3 bin=0118A1023_Clostridium_difficile_28_5          75
## 4 bin=0118A1023_Enterococcus_faecalis_36_53         74
## 5 bin=0118A1023_UNK                                  4
```

There are 71 unique single copy core genes that were identified among these contigs, and 4 of the genomes contain a little over 71 indicating some redundancy of these genomes. The unknown genome only contains 4 single copy core genes identified in this pipeline. This is a good check, as there are 71 single copy markers in the Bacteria_71 set, and they were all found within these sets of contigs.

I then imported the HMM hits for the Archaea_76 set just to check:

```
hmm_hits_archaea <- read.table("results/hmm-hits-archaea.txt", header=TRUE, sep="\t")

hmm_table_archaea <- left_join(hmm_hits_archaea, genes_table)
```

```
## Joining, by = "gene_callers_id"
```

```
hmm_table_archaea %>%
  select(gene_name) %>%
  unique() %>%
  count()
```

```
##    n
## 1 42
```

```
hmm_table_archaea %>%
  group_by(bin) %>%
  unique() %>%
  count()
```

```
## # A tibble: 5 x 2
## # Groups:   bin [5]
##   bin                                                n
##   <chr>                                          <int>
## 1 bin=0118A1023_Bacteroides_vulgatus-like_42_520    35
## 2 bin=0118A1023_Citrobacter_koseri_53_430           35
## 3 bin=0118A1023_Clostridium_difficile_28_5          42
## 4 bin=0118A1023_Enterococcus_faecalis_36_53         39
## 5 bin=0118A1023_UNK                                  3
```

Of the 76 total HMM markers for the archaea set, there are only 42 unique hits among these contigs. The count for the unknown bin for the archaeal HMM hits are low as well (3 archaeal hits vs 4 bacterial hits), and are generally low for all genomes, so we can be confident that these aren't archaeal genomes.

I then wanted to compare the classification obtained through the Anvi'o workflow based on the taxonomy of the Ribosomal L2 gene and the given classification in the contigs file.

```
# import the taxonomy results
tax_table <- read.table("results/infant-gut-tax.txt", header=TRUE, sep="\t") %>%
  mutate(gene_callers_id = gsub("Ribosomal_L2_", "", scg_name)) %>%
  unite("taxonomy", 3:9, sep=";")

tax_table$gene_callers_id <- as.integer(tax_table$gene_callers_id)

# join with the gene calls table for the bin
left_join(tax_table, genes_table) %>%
  select(bin, taxonomy)
```

```
## Joining, by = "gene_callers_id"
```

```
##                                                bin
## 1       bin=0118A1023_Clostridium_difficile_28_5
## 2        bin=0118A1023_Citrobacter_koseri_53_430
## 3 bin=0118A1023_Bacteroides_vulgatus-like_42_520
## 4      bin=0118A1023_Enterococcus_faecalis_36_53
## 5                             bin=0118A1023_UNK
##
## 1 Bacteria;Firmicutes;Clostridia;Peptostreptococcales;Peptostreptococcaceae;Clostridioides;Clostridi
## 2                         Bacteria;Proteobacteria;Gammaproteobacteria;Enterobacterales;Enterobacteria
## 3                   Bacteria;Bacteroidota;Bacteroidia;Bacteroidales;Bacteroidaceae;Phocaeicola;Phocae
## 4                    Bacteria;Firmicutes;Bacilli;Lactobacillales;Enterococcaceae;Enterococcus;Enteroc
## 5 Bacteria;Desulfobacterota;Desulfovibrionia;Desulfovibrionales;Desulfovibrionaceae;Bilophila;Bilophi
```

Comparing the results of the classification from the Ribosomal L2 hits with the preliminary classifications given in the contigs file, the classifications match very well accounting that there are taxonomical differences in species names such as "Bacteroides vulgatus" and "Phocaeicola vulgatus" because of differences between the NCBI and GTDB taxonomy systems. Additionally, based on the classification of the Ribosomal L2 gene within the `0118A1023_UNK` bin that did not contain a given classification in the contigs file, it is classified as `Bilophila wadsworthia` within the Desulfobacterota, which in the NCBI taxonomy roughly equates to the Deltaprotoebacteria which is split into multiple phyla in the GTDB.

This quick sanity check ensures that 1) There are indeed approximately 5 bacterial “species” based on the single copy core gene analysis since only 5 Ribosomal L2 genes were identified and 2) The classification of those ribosomal L2 genes matches the classifications given for the bins in the contigs file. Additionally, this analysis provided a putative classification for the unknown bin. Furthermore, by running the `Bacteria_71` HMM collection on the contigs, we can tell that the 4 genomes besides the unknown/Desulfobacterota bin contain a high number of single copy core genes and are likely of higher quality. For now for downstream annotation and analysis we will keep the unknown/Desulfobacterota bin and take the results into account for the possible lower quality of this bin and can remove those results later if we aren’t confident in them.

The correct and accurate analysis for this set of contigs would be to run GTDB-tk to confirm the classification and run checkM to assess completeness and contamination. However each of these tools requires a higher memory allocation and resources than quick checks with Anvi’o and for a quick spot check this analysis will suffice.

Functional Annotation

Since the anvi’o pipeline already predicted ORFs with prodigal, we can export the proteins with `anvi-get-sequences-for-gene-calls -c contigs.db --get-aa-sequences -o infant-gut-proteins.faa` and then perform functional annotation. First I will annotate with KofamKOALA, which will annotate all proteins based on the KEGG database. This is to generate general annotations mainly for the bacterial species for a general overview of functions and to inspect gene neighborhoods in which mobile regions are found. Since the KofamKOALA online portal only takes 10,000 proteins for a single job, I split the FASTA file into two containing each about 8,500 proteins for all genomes, and then combined the results back together. The KofamKOALA pipeline can also be run on a personal laptop but the database is quite large and requires more threads, and can also be installed on a larger compute cluster for larger sets of annotation searches. The KofamKOALA pipeline then returns all annotation hits based on HMMs for each gene, and then hits can be filtered for significant annotations based on those that were found above the given threshold cutoff (TC) for the HMM. I then sorted the file so that only the top hit for that protein was kept so there wasn’t duplicate hits for a single protein to deal with using `sort -u -k1,1 sig_kofam_annotations_modf.txt > sig_kofam_annotations_modf_nodup.txt`.

```
kofam_annotations <- read.table("results/sig_kofam_annotations_modf_nodup.txt", col.names = c("gene_cal

kofam_annotations_bins <- left_join(kofam_annotations, genes_table)
```

```
## Joining, by = "gene_callers_id"
```

```
kofam_annotations_bins %>%
  group_by(bin) %>%
  count() %>%
  arrange(desc(n))
```

```
## # A tibble: 7 x 2
## # Groups:   bin [7]
##   bin                                             n
##   <chr>                                       <int>
## 1 bin=0118A1023_Citrobacter_koseri_53_430      1552
## 2 bin=0118A1023_Clostridium_difficile_28_5     1157
## 3 bin=0118A1023_Bacteroides_vulgatus-like_42_520  541
## 4 bin=0118A1023_Enterococcus_faecalis_36_53     500
## 5 bin=0118A1023_UNK                              32
## 6 bin=0118A1023_Proteobacteria_phage              2
## 7 bin=0118A1023_Enterococcus_faecalis_phage       1
```

From this we can see that the KofamKOALA annotation pipeline annotated above 1000 proteins in the Citrobacter and C. diff genomes, only about 500 in the B. vulgatis and E. faecalis genomes, and very few proteins in the unknown/Desulfobacterota genomes and the two phage genomes. With the combined results of the low single copy core genes detected in the unknown/Desulfobaterota genome and the very few annotations detected, I would filter out these scaffolds from downstream results since this is likely contamination or a genome that was not assembled well.

I then used the mobileOG database to annotate mobile elements in all genomes, which is described in the *bioRxiv* preprint by Brown et al. 2021. This database is an extensive collection of mobile elements that was created by merging 7 other publicly available databases:

1. ICEBerg 2.0 (ICEs, AICEs, CIMEs, IMEs)
2. COMPASS (plasmids)
3. NCBI Plasmid RefSeq (plasmids)
4. Gut Phage Database (phages)
5. ACLAME (various)
6. immedb (integrative elements)
7. Prokaryotic viral orthologous groups (pVOG)

Following their instructions in the mobileOG Github repository, I first made a DIAMOND database of the mobileOG collection of proteins:

```
diamond makedb --in mobileOG-db-beatrix-1.5.All.faa -d mobileOG-db-beatrix-1.5.All.dmnd
```

Then compared the infant gut proteins to the mobileOG database with:

```
diamond blastp -q ../raw_data/infant-gut-proteins.faa --db mobileOG-db-beatrix-1.5.All.dmnd
-o ../results/infant-gut-mobileOG.tsv -k 15 -e 1e-20 --query-cover 90 --id 90
```

The mobileOG database is provided as a large CSV which I have reformatted as a smaller TSV containing the necessary columns to obtain annotation information. The functions for each mobileOG database hit can then be matched in the output file. The output file for the infant gut proteins produced 4,392 annotations that hit above the cutoffs, from which I filtered the results with `sort -u -k1,1 infant-gut-mobileOG.tsv` to only pick the highest hit for a protein so a given protein didn't have duplicate annotations to simplify the analysis.

```r
# read in the mobileOGdb file
mobileOGdb <- as.data.frame(readLines("beatrix-1-5_v1_all/mobileOG-db-beatrix-1.5.All.modf.tsv"))
colnames(mobileOGdb) <- c("Description")
mobileOGdb_table <- mobileOGdb %>%
  separate(Description, into=c("Entry", "Cluster", "Name", "Annotation"), sep="\t")

# import the DIAMOND tsv output for infant gut proteins against mobileOGdb
infant_gut_mobileOGdb <- read.table("results/infant-gut-mobileOG.filtered.tsv", col.names = c("gene_cal
  separate(mobileOGdb_hit, into=c("Entry"), sep="\\|")
```

```
## Warning: Expected 1 pieces. Additional pieces discarded in 902 rows [1, 2, 3, 4,
## 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```r
# join the infant gut hits with the database hits
infant_gut_mobileOGdb_hits <- left_join(infant_gut_mobileOGdb, mobileOGdb_table)
```

```
## Joining, by = "Entry"
```

```r
# join with the genes/bins table and filter out annotations with NA descriptions
infant_gut_mobileOGdb_hits_bins <- left_join(infant_gut_mobileOGdb_hits, genes_table) %>%
  filter(Name != "NA:Keyword") %>%
  filter(bin != "bin=0118A1023_UNK") # also filter out any hits to the unknown bin scaffolds
```

```
## Joining, by = "gene_callers_id"
```

With the above filtering steps and removing hits to the "Unknown" bin scaffolds since the single copy core analysis and preliminary annotations showed this is probably a low quality bin, 524 mobile elements were annotated among the scaffolds. Then checking the annotation breakdown for each bin:

```
infant_gut_mobileOGdb_hits_bins %>%
  group_by(bin) %>%
  count() %>%
  arrange(desc(n))
```

```
## # A tibble: 7 x 2
## # Groups:   bin [7]
##   bin                                              n
##   <chr>                                        <int>
## 1 bin=0118A1023_Bacteroides_vulgatus-like_42_520   191
## 2 bin=0118A1023_Citrobacter_koseri_53_430          162
## 3 bin=0118A1023_Enterococcus_faecalis_36_53         97
## 4 bin=0118A1023_Clostridium_difficile_28_5          56
## 5 bin=0118A1023_Enterococcus_faecalis_phage          7
## 6 bin=0118A1023_Klebsiella_pneumoniae_plasmid        3
## 7 bin=0118A1023_Proteobacteria_phage                 1
```

The B. vulgatis genome had nearly 200 mobile elements annotated, the C. koseri genome 162, the E. faecalis 97, the C. diff 56, and the E. faecalis phage 7 elements annotated. The Klebsiella plasmid had very few mobile elements annotated, and the Proteobacteria phage only 1. However this pipeline picked up more annotations in total for the E. faecalis phage, as the KofamKOALA pipeline didn't functionally annotate any proteins above the threshold cutoffs for this phage. The pipeline could be altered by adjusting the e-values, percent identity, and coverage cutoffs for the DIAMOND blastp job. But for a first glance these higher cutoff values and filtering parameters will suffice.

Exploring Transposases

```
infant_gut_mobileOGdb_hits_bins %>%
  filter(str_detect(Annotation, 'Transposase|transposase|transposon')) %>%
  group_by(bin) %>%
  count() %>%
  arrange(desc(n))
```

```
## # A tibble: 4 x 2
## # Groups:   bin [4]
##   bin                                              n
##   <chr>                                        <int>
## 1 bin=0118A1023_Bacteroides_vulgatus-like_42_520    19
## 2 bin=0118A1023_Clostridium_difficile_28_5           7
## 3 bin=0118A1023_Enterococcus_faecalis_36_53          6
## 4 bin=0118A1023_Citrobacter_koseri_53_430            4
```

Based on these results, there are ~20 transposases in the B. vulgatis genome, and a handful in the other 3 bacterial species.

```
transposase_table <- infant_gut_mobileOGdb_hits_bins %>%
  filter(str_detect(Annotation, 'Transposase|transposase|transposon')) %>%
  select(gene_callers_id, Entry, Name, Annotation, contig, bin)

transposase_gene_ids <- transposase_table %>%
  pull(gene_callers_id)

write.table(transposase_gene_ids, "results/transposase_gene_ids.txt", col.names = FALSE, quote = FALSE,
```

From the anvi'o pipeline, we can also export the gene calls from prodigal to have the ORFs to match up to the proteins for performing BLAST comparisons of regions where transposons/integrons are located with `anvi-get-sequences-for-gene-calls -c contigs.db -o infant-gut-orfs.ffn`. Then we can locate the gene caller ID for each transposase and investigate if the genes flanking the transposase shares homology with either species in the infant gut community.

From the list of gene IDs for the transposases, we first retrieve the coding regions for the genes surrounding each transposase. For this exercise 10-20 genes will suffice to see if regions in other species match homology to a given transposase neighborhood. For each transposon gene ID exported in the `transposase_gene_ids.txt` file, we can retrieve the protein sequence for that transposase and the proteins for the genes that are 10 genes upstream and downstream of the transposase to see if there is homology to regions in other genomes. Then we create DIAMOND databases for each set of transposase neighborhoods, and then search those databases against the full set of proteins in the infant gut metagenome. We can get a quick look at this to see if there are any low-hanging fruit with a couple bash for loops:

First retrieve the protein sequences for the transposases and surrounding proteins with `for line in $(cat transposase_gene_ids.txt); do grep -A 21 -B 21 -w $line ../raw_data/infant-gut-proteins.faa > transposases/transposases-$line.txt; done`

Then create DIAMOND databases for each set of proteins with `for file in *.txt; do name=$(basename $file .txt); diamond makedb --in $file --d $name.dmnd; done`

Then for each DIAMOND database search against the infant gut proteins. For this since we are searching against proteins the search parameters will be a little relaxed. We could also use conventional BLAST to perform blastn searches of the coding regions of the transposase neighborhoods, but DIAMOND searches against protein databases and will suffice for a quick look. To do this for each DIAMOND database: `for file in *.dmnd; do name=$(basename $file .dmnd); diamond blastp -q ../../raw_data/infant-gut-proteins.faa --db $file -o $name.tsv -k 15 --id 70 --query-cover 85 -e 1e-20; done`

Then I inspected the output files for proteins that were in the genomes of species other than the query transposase. Most of the results identified homologous regions of the same genome that also had transposase hits, such as a C. diff transposase query hitting other transposases and homologous gene neighborhoods within the C. diff genome. I was specifically interested in inter-species signatures of transposase-mediated horizontal gene transfer, so I was looking for regions in other species.

I found one example of a C. diff transposase (gene caller ID 3413, annotated as a Transposase) and the surrounding gene neighborhood matching homology to a transposase and surrounding gene neighborhood in E. faecalis (transposase gene caller ID 8417). I then retrieved the coding regions surrounding these transposases to inspect the annotations and homology of these gene neighborhoods further.

To prepare the files, first I retrieved the coding regions of the 10 genes downstream and upstream of the transposase of interest in the E. facecalis and C. diff genomes:

```
grep -A 21 -B 20 -w '8147' infant-gut-orfs.ffn > ../results/Efaecalis-transposon-gene-neighborhood.ffn
```

```
grep -A 21 -B 20 -w '3413' infant-gut-orfs.ffn > ../results/Cdiff-3413-transp-neighborhood.ffn
```

I'm using the python program `clinker` to analyze gene clusters. I installed this locally in a conda environment using `conda create -n clinker -c conda-forge -c bioconda clinker-py` and activated the environment. For this program, the coding regions need to correspond to a GFF3 file, which I can retrieve from the Anvi'o gene calls. With the corresponding GFF3 and FASTA files for the scaffolds on which these sequences are on, then we run `clinker *.gff` and this program will perform clustering and alignment of the protein sequences. It gives arbitrary string names to the names of genes with an alignment greater than 30%, and this aligns well with the previous DIAMOND blastp results:

```
Query                                 Target                                Identity  Similarity
331e65bf-8771-4662-893f-53e0b3dd9c86  03fd2a86-de61-4ce7-8454-dc0b6f841b55  0.62      0.75
2c673e56-4254-4d74-a59e-0bb29786dd98  eff1c082-5f9d-40e9-9206-e38f3083afa9  0.78      0.90
```

4a2a10c0-67d1-4828-ad08-fc3e4efef45a	b17ed0ec-bb3c-4f03-a501-46797e4d67a1	0.65	0.79
19a1db67-2887-4575-8c7a-83ec6cd085de	4ae47710-1c44-4456-a257-406afd9c4782	0.45	0.55
12c261ca-215a-408d-913b-a80154151620	bc13eb41-b909-43d3-b786-443f717776e0	0.71	0.81
99f8d0b1-1967-4a10-97b5-7cdf8bd27a93	c885d235-c9de-4ddf-9a0e-cf8374be2794	0.52	0.69
35c1f89c-a77d-4f27-b214-c9e3eee4bf66	435c76b3-b64e-461a-95d1-c126992ca69a	0.66	0.80
59a8d91d-7591-4abe-b0da-de4abaa05600	283e85cf-f0bb-4510-aede-4bc74d4ff1b4	0.50	0.62
ffa0a000-abc2-4f8c-b73f-1bf006299def	ceac2ef0-e4a0-44e7-9f0c-2a48a8c5270b	0.81	0.90
72f03c29-170a-4b63-98a1-a99b74a830d5	fc34858b-8dfa-4b35-9a2a-c2f0a1b7f60a	0.71	0.84
5b3dc4ae-6b41-49d9-a560-27c02802b00e	e0e57067-9a2f-4f6f-b025-c1f1c393aba9	0.62	0.78
b7865adb-fe84-4265-947f-7e0fec681ae0	538466c3-93f3-48ba-a1b2-de963161d25e	0.86	0.96
29721566-b1fa-4dcc-874e-87bccb7e896d	7eeab21b-7fc2-4be4-9aec-9dcaa0eb8f64	0.60	0.70
bb70f340-eb29-4050-b3a4-d2aab2a57a99	9777b854-3acc-4e18-86a1-8ba515c61332	0.66	0.79
123469d5-945b-40e0-9b17-96e7658221d2	a9297182-60db-45b1-8926-569332ea02f5	0.77	0.87
e4ebf843-652f-4a4b-b58b-dae95914f41e	a009c36d-fd89-4926-a80b-8583fa6704be	0.80	0.90

I exported the SVG output into Illustrator so I could focus on the region of interest since clinker requires inputting entire scaffolds for comparison, and only one small region of one of the scaffolds shares similarity with the other scaffold. Then I looked at the KofamKOALA and mobileOG annotations for these gene neighborhoods to see if these databases could help with functional annotation as well as having the homology information.