Script P11: Functional Analysis

HANNIGAN GD, GRICE EA, ET AL.

Abstract

This protocol provides method for KEGG Analysis, Gene Ontology Analysis, and Operational Protein Family Analysis. Based on the methods from the following publication:

Hannigan, Geoffrey D., et al. "The Human Skin Double-Stranded DNA Virome: Topographical and Temporal Diversity, Genetic Enrichment, and Dynamic Associations with the Host Microbiome." *mBio* 6.5 (2015): e01578-15.

Citation: HANNIGAN GD, GRICE EA, ET AL. Script P11: Functional Analysis. **protocols.io** dx.doi.org/10.17504/protocols.io.ehubb6w
Published: 10 Mar 2016

Guidelines

- Output
  - For KEGG analysis:
    - KEGG_humann/04b-hit-keg-mpm-cop-nul-nve-nve-skinmet.txt
    - KEGG_humann/04b-hit-keg-mpm-cop-nul-nve-nve-virome.txt
  - For GOEAST analysis
    - Step one input files in IntermediateOutput/GOEAST/
      - sebaceous_GO_ids_formatted.txt
      - moist_GO_ids_formatted.txt
      - rimoist_GO_ids_formatted.txt
    - Step two input files in IntermediateOutput/GOEAST/
      - sebaceous_skinmet_uniprot_blastx_unique_probes.txt & sebaceous_virome_uniprot_blastx_unique_probes.txt
      - moist_skinmet_uniprot_blastx_unique_probes.txt & moist_virome_uniprot_blastx_unique_probes.txt
      - rimoist_skinmet_uniprot_blastx_unique_probes.txt & rimoist_virome_uniprot_blastx_unique_probes.txt
    - GOEAST output files (for visualization with Multi-GOEAST) in IntermediateOutput/GOEAST/output/
      - result_contigs-seb-skinmet.txt & result_contigs-seb-virome.txt
      - result_contigs-moist-skinmet.txt & result_contigs-moist-virome.txt
      - result_contigs-rimoist-skinmet.txt & result_contigs-rimoist-virome.txt
  - For Operational Protein Family analysis
    - ./protein_cluster_rel_abund_table/contig_otu_table_transposed.txt
- R scripts: R12 and R13

Before start

Perl scripts and other supplemental information available at:

https://figshare.com/articles/The_Human_Skin_dsDNA_Virome_Topographical_and_Temporal_Diversity_Genetic_Enrichment_and_Dynamic_Associations_with_the_Host_Microbiome/1281248

Protocol

KEGG Analysis

Step 1.

In order to perform KEGG functional analysis, we had to blast our samples against the KEGG database and then use the program HUMAnN to analyze coverage/abundance of different modules/pathways. First we downloaded the program, HUMAnN. Then, since blastx is very time intensive, we first subsampled our sequences to 10,000 using the seqtk toolkit.

cmd COMMAND

```
mkdir ./HUMAnN_10k
mkdir ./HUMAnN_10k/subsampled_input

for file in $(ls ./MetaPhlAn_trimmed/input_reads); do
    seqtk sample ./MetaPhlAn_trimmed/input_reads/$file 10000 > ./HUMAnN_10k/subsampled_inpu
t/${file/fastq/_subsampled.fastq}
    seqtk seq -
a ./HUMAnN_10k/subsampled_input/${file/fastq/_subsampled.fastq} > ./HUMAnN_10k/subsampled_i
nput/${file/.fastq/_subsampled.fa}
done

cd ./HUMAnN_10k/subsampled_input/
rm *.fastq
```

KEGG Analysis

Step 2.

Some of our samples had fewer than 10,000 sequences, and were removed from further analysis. This included samples MG100184 and MG100755 for the metagenome and samples MG100099, MG100105, MG100416, MG100497, and MG100613 for the virome. Next, we blasted our samples against the KEGG database. We used an e-value of 1e-10 for the whole metagenome samples and 1e-5 for the virome samples.

SOFTWARE PACKAGE (Unix)

BLAST Toolkit, 2.2.0

NCBI
ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

cmd COMMAND

```
run.blast.parallel.for.kegg.HUMAnN () {
     blastx -query ./HUMAnN_10k/subsampled_input/${1}.fa -
out ./HUMAnN_10k/humann-0.99/input/${1}.blastx -db references/kegg/kegg.reduced.fasta -
outfmt 6 -num_threads 2 -evalue 1e-10
}

export -f run.blast.parallel.for.kegg.HUMAnN
ls ./HUMAnN_10k/subsampled_input/ | sed -e 's/^.*\/.*\///g' | sed 's/\.fa//g' | xargs -
I {} --max-procs=128 sh -c '"run.blast.parallel.for.kegg.HUMAnN {}"'
```

KEGG Analysis

Step 3.
Finally, we are ready to run HUMAnN

SOFTWARE PACKAGE (Unix)
HUMAnN: The HMP Unified Metabolic Analysis Network, 0.99
Curtis Huttenhower
COMMAND
```
cd ./HUMAnN_10k/humann-0.99/
scons
```

NOTES
Geoffrey Hannigan 04 Feb 2016
We performed downstream analyses in R in order to generate Supplemental Figure 12.

Gene Ontology Analysis
Step 4.
In order to perform gene ontology analysis, we used the online Gene Ontology Enrichment Analysis Software Toolkit. Under GOEAST Advance, we used Customized-GOEAST. This treats the data like it is a microarray. In our case, the microarray probes are the open reading frames (ORFs) from our contigs.

Gene Ontology Analysis
Step 5.
In order to compare the virome and metagenome samples, we first grouped them by microenvironment (sebaceous, moist, and intermittently moist). For each microenvironment, we annotated all of the virome ORFs and metagenome ORFs with GO terms by blasting the contigs against the UniProt SwissProt database and mapping the SwissProt hits to GO IDs.

Gene Ontology Analysis
Step 6.
For step one of Customized-GOEAST, our annotation file contained all of the annotated contigs from both the virome and metagenome samples. We ran this analysis twice, first using only the virome contigs as our input list of probes and then using only the metagenome contigs as our list of probes for step 2 of Customized-GOEAST. We used the default parameter settings in the optional section.

Gene Ontology Analysis
Step 7.
Since we need to map our ORFs (detected by glimmer3) to GO IDs, we first subsampled (to 1000 sequences) and blasted the ORFs against the SwissProt database. We do not remove any samples with fewer than 1000 sequences.

COMMAND
```
cd ./glimmer3
mkdir orf_uniprot_blast_swissprot_sub

runblastxorfs () {
     # Remove block formatting
     fasta_formatter -
w 0 < ./glimmer3/output/${1}.genes > ./glimmer3/output/${1}_no_block.genes
     # Subsample
     seqtk sample ./glimmer3/output/${1}_no_block.genes 1000 > ./glimmer3/orf_uniprot_blast
_swissprot_sub/${1}_sub.genes
     # Blast
     blastx -query ./glimmer3/orf_uniprot_blast_swissprot_sub/${1}_sub.genes -
out ./glimmer3/orf_uniprot_blast_swissprot_sub/${1}_orf_uniprot_blastx.txt -
db references/SwissProt/uniprot_sprot.fasta -outfmt 6 -num_threads 16 -evalue 1e-10 -
max_target_seqs 1
}
```

```
export -f runblastxorfs
ls ./glimmer3/output/*_no_block.genes | sed -
e 's/^.*\/.*\///g' | sed 's/\_no_block.genes//g' | xargs -I {} --max-procs=128 sh -
c 'runblastxorfs {}'
```

Gene Ontology Analysis

Step 8.

Now that we have UniProt hits for our open reading frames, we had to find a way to match our UniProt hits to GO IDs. We downloaded and formatted the following reference file for our use:

cmd COMMAND

```
wget ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/UNIPROT/gp_association.goa_uniprot.gz
gunzip gp_association.goa_uniprot.gz
tail -
n +24 gp_association.goa_uniprot | awk '{print $2,$4}' > gp_association.goa_uniprot.goeast_
formatted
```

Gene Ontology Analysis

Step 9.

Next, we combine samples by their microenvironment. We broke the mapping file up into two separate mapping files- one for the whole metagenome and one for the virome. In this new mapping file, column 1 contained the sample ID and column 5 contained the site symbol.

cmd COMMAND

```
# group by bodysite
cat SkinMet_001_metadata_subset.tsv | awk '$5=="Fh" || $5=="Ra" || $5=="Sc" {print $1}' > s
ebaceous_skinmet_samples.txt
cat SkinMet_001_metadata_subset.tsv | awk '$5=="Ax" || $5=="Um" || $5=="Tw" {print $1}' > m
oist_skinmet_samples.txt
cat SkinMet_001_metadata_subset.tsv | awk '$5=="Ac" || $5=="Pa" {print $1}' > rimoist_skinm
et_samples.txt
cat Virome_001_metadata_subset.tsv | awk '$5=="Fh" || $5=="Ra" || $5=="Sc" {print $1}' > se
baceous_virome_samples.txt
cat Virome_001_metadata_subset.tsv | awk '$5=="Ax" || $5=="Um" || $5=="Tw" {print $1}' > mo
ist_virome_samples.txt
cat Virome_001_metadata_subset.tsv | awk '$5=="Ac" || $5=="Pa" {print $1}' > rimoist_virome
_samples.txt

for file in $(ls *_samples.txt); do
    name=${file/samples.txt/uniprot_blastx.txt}
    for sample in $(cat $file); do
        cat ${sample}_orf_uniprot_blastx.txt >> $name
    done
    awk '{print $1"\t"$2}' $name | sed 's/.orf[0-9]*//g' | sed 's/sp|//g' | sed 's/|.*//g'
| sort -u > ${name/.txt/_formatted_unique.txt}
    cut -f 1 ${name/.txt/_formatted_unique.txt} | sort -u > ${name/.txt/_unique_probes.txt}
done
```

Gene Ontology Analysis

Step 10.

We annotate the UniProt hits with GO IDs and format them for input as step one of Customized-GOEAST.

SOFTWARE PACKAGE (Unix)

GOEAST (Gene Ontology Enrichment Analysis Software Toolkit)

Institute of Genetics and Developmental Biology

cmd COMMAND

```
cat sebaceous_skinmet_uniprot_blastx_unique_probes.txt sebaceous_virome_uniprot_blastx_uniq
ue_probes.txt > sebaceous_probes.txt
cat sebaceous_skinmet_uniprot_blastx_formatted_unique.txt sebaceous_virome_uniprot_blastx_f
ormatted_unique.txt | sort -u -k 2 > sebaceous_hits.txt
```

```
join -1 2 -2 1 sebaceous_hits.txt gp_association.goa_uniprot.goeast_formatted > sebaceous_G
O_ids.txt
awk '{print $2"\t"$3}' sebaceous_GO_ids.txt | sort -u > sebaceous_GO_ids_unique.txt

for line in $(cat sebaceous_probes.txt); do
    grep -w "$line" sebaceous_GO_ids_unique.txt | perl -
pe 's/'$line'/\/\//g' | tr '\t' ' ' | tr '\n' ' ' | sed 's/^\/\///g' | sed 's/$/\n/g' | sed
 "s/^/$line\t/g" >> sebaceous_GO_ids_formatted.txt
done

cat moist_skinmet_uniprot_blastx_unique_probes.txt moist_virome_uniprot_blastx_unique_probe
s.txt > moist_probes.txt
cat moist_skinmet_uniprot_blastx_formatted_unique.txt moist_virome_uniprot_blastx_formatted
_unique.txt | sort -u -k 2 > moist_hits.txt
join -1 2 -2 1 moist_hits.txt gp_association.goa_uniprot.goeast_formatted > moist_GO_ids.tx
t
awk '{print $2"\t"$3}' moist_GO_ids.txt | sort -u > moist_GO_ids_unique.txt

for line in $(cat moist_probes.txt); do
    grep -w "$line" moist_GO_ids_unique.txt | perl -
pe 's/'$line'/\/\//g' | tr '\t' ' ' | tr '\n' ' ' | sed 's/^\/\///g' | sed 's/$/\n/g' | sed
 "s/^/$line\t/g" >> moist_GO_ids_formatted.txt
done

cat rimoist_skinmet_uniprot_blastx_unique_probes.txt rimoist_virome_uniprot_blastx_unique_p
robes.txt > rimoist_probes.txt
cat rimoist_skinmet_uniprot_blastx_formatted_unique.txt rimoist_virome_uniprot_blastx_forma
tted_unique.txt | sort -u -k 2 > rimoist_hits.txt
join -1 2 -2 1 rimoist_hits.txt gp_association.goa_uniprot.goeast_formatted > rimoist_GO_id
s.txt
awk '{print $2"\t"$3}' rimoist_GO_ids.txt | sort -u > rimoist_GO_ids_unique.txt

for line in $(cat rimoist_probes.txt); do
    grep -w "$line" rimoist_GO_ids_unique.txt | perl -
pe 's/'$line'/\/\//g' | tr '\t' ' ' | tr '\n' ' ' | sed 's/^\/\///g' | sed 's/$/\n/g' | sed
 "s/^/$line\t/g" >> rimoist_GO_ids_formatted.txt
done
```

NOTES

Geoffrey Hannigan 04 Feb 2016

For Figure S10, we pulled out a section of interest from the Biological Process graph from the sebaceous virome samples.

Geoffrey Hannigan 04 Feb 2016

For Figure S10, we pulled out a section of interest from the Biological Process graph from the sebaceous virome samples.

Operational Protein Family Analysis

Step 11.

We also calculated operational protein families to assess core and flexible OPFs, potential auxiliary metabolic genes, and the functional diversity. The OPFs are defined below. Start by clustering the predicted ORFs using Uclust to remove the redundant sequences and pull out representative sequences.

cmd COMMAND

```
mkdir ./define_core_protein_clusters
echo Sorting fasta with uclust...
uclust --sort ./glimmer3/output/Contigs_no_block_with_names_glimmer_output_final.fa --
output ./define_core_protein_clusters/glimmer_final_sorted.fa
```

Operational Protein Family Analysis

Step 12.
Use the sorted file for de novo clustering.

cmd COMMAND
```
echo Using uclust for sequence clustering...
uclust --input ./define_core_protein_clusters/glimmer_final_sorted.fa --
uc ./define_core_protein_clusters/protein_clusters_uclust.txt --id 0.75
```
Operational Protein Family Analysis

Step 13.
Convert output to fasta.

cmd COMMAND
```
echo Converting uclust output to fasta...
uclust --uc2fasta ./define_core_protein_clusters/protein_clusters_uclust.txt --
input ./define_core_protein_clusters/glimmer_final_sorted.fa --
output ./define_core_protein_clusters/protein_clusters_with_repeats.fa
```
Operational Protein Family Analysis

Step 14.
Pull out the clustered sequences which have representative seqs in their sequences.

cmd COMMAND
```
echo Pulling out representative seqs from uclust fasta output...
grep -A 1 "*" ./define_core_protein_clusters/protein_clusters_with_repeats.fa | grep -v -
P ^-- > ./define_core_protein_clusters/protein_clusters.fa
```
Operational Protein Family Analysis

Step 15.
Get the length information for each protein cluster representative sequence.

cmd COMMAND
```
awk 'NR % 2 {printf $0"\t"} !(NR % 2) {print length($0)}' ./define_core_protein_clusters/pr
otein_clusters.fa > ./define_core_protein_clusters/protein_cluster_length.txt
sed 's/>//g' ./define_core_protein_clusters/protein_cluster_length.txt > ./define_core_prot
ein_clusters/protein_cluster_length_no_greater_sign.txt
```

NOTES

Geoffrey Hannigan 04 Feb 2016

Use the defined operational protein families to generate a OPF relative abundance table which can be used in R for diversity, definition of core OPFs, etc.

Operational Protein Family Analysis

Step 16.
Run bowtie2 of the background cleaned samples against the contig reference database. First build bowtie reference of the OPFs.

cmd COMMAND
```
mkdir ./protein_cluster_rel_abund_table
bowtie2-build -
f ./define_core_protein_clusters/protein_clusters.fa ./protein_cluster_rel_abund_table/bowt
ie2_contig_build
```
Operational Protein Family Analysis

Step 17.
Map the sequences to the OPFS.

cmd COMMAND
```
mkdir ./protein_cluster_rel_abund_table/bowtie2_neg_cleaned_hits
run_bowtie2_against_contigs () {
if [ -f /etc/profile.d/modules.sh ]; then
       source /etc/profile.d/modules.sh
fi
module load bowtie2-2.1.0
bowtie2 -x ./protein_cluster_rel_abund_table/bowtie2_contig_build -
f ./negative_clean_seqs/$1 -
S ./protein_cluster_rel_abund_table/bowtie2_neg_cleaned_hits/$1 -L 25 -N 1
}
export -f run_bowtie2_against_contigs
ls ./negative_clean_seqs/ | xargs -I {} --max-procs=128 sh -
c 'run_bowtie2_against_contigs {}"'
```

Operational Protein Family Analysis

Step 18.

Rename the files to be .sam files.

cmd COMMAND

```
for file in $(ls ./protein_cluster_rel_abund_table/bowtie2_neg_cleaned_hits); do
    mv ./protein_cluster_rel_abund_table/bowtie2_neg_cleaned_hits/"${file}" ./protein_clust
er_rel_abund_table/bowtie2_neg_cleaned_hits/"${file/%.fa/.sam}"
done
```

Operational Protein Family Analysis

Step 19.

Calculate the hit abundances from bowtie2 using estimation perl script.

cmd COMMAND

```
mkdir ./protein_cluster_rel_abund_table/abundance_from_sam
for file in $(ls ./protein_cluster_rel_abund_table/bowtie2_neg_cleaned_hits); do
    perl calculate_abundance_from_sam.pl ./protein_cluster_rel_abund_table/bowtie2_neg_clea
ned_hits/${file} ./protein_cluster_rel_abund_table/abundance_from_sam/${file}
done
```

NOTES

Geoffrey Hannigan 04 Feb 2016

Perl script available here.

Operational Protein Family Analysis

Step 20.

Rename the sam files to text files.

cmd COMMAND

```
for file in $(ls ./protein_cluster_rel_abund_table/abundance_from_sam); do
    mv ./protein_cluster_rel_abund_table/abundance_from_sam/"${file}" ./protein_cluster_rel
_abund_table/abundance_from_sam/"${file/%.sam/.txt}"
done
```

Operational Protein Family Analysis

Step 21.

Add in OPF length information using awk and calculate RPKM.

cmd COMMAND

```
mkdir ./protein_cluster_rel_abund_table/abundance_with_length
for file in $(ls ./protein_cluster_rel_abund_table/abundance_from_sam); do
    export SUM=$(awk '{ SUM += $2 } END { print SUM }' ./protein_cluster_rel_abund_table/ab
undance_from_sam/${file})
```

Operational Protein Family Analysis

Step 22.

Add an echo of the sum value to confirm that the sum is being calculated.

cmd COMMAND

```
echo Sum is $SUM
    awk --
assign=sum=$SUM 'FNR==NR { a[$1]=$2; next } $1 in a { print $1"\t"$2"\t"a[$1]"\t"$2*1000000
000/(a[$1]*sum) }' ./define_core_protein_clusters/protein_cluster_length_no_greater_sign.tx
t ./protein_cluster_rel_abund_table/abundance_from_sam/${file} > ./protein_cluster_rel_abun
```

```
d_table/abundance_with_length/${file}
done
```

Operational Protein Family Analysis

Step 23.

Get columns of only the contig IDs and the normalized RPKM abundance counts and add sample name to top of column for when this is used in distance matrix calculations,

cmd COMMAND

```
mkdir ./protein_cluster_rel_abund_table/abundance_RPKM_with_only_contig_id
for file in $(ls ./protein_cluster_rel_abund_table/abundance_with_length); do
    NAME=$(echo ${file} | sed 's/_R1\.txt//')
    echo Name is $NAME
    cut -
f 1,4 ./protein_cluster_rel_abund_table/abundance_with_length/${file} | sed "1 s/^/Contig_I
D\t${NAME}\n/" > ./protein_cluster_rel_abund_table/abundance_RPKM_with_only_contig_id/${fil
e}
done
```

Operational Protein Family Analysis

Step 24.

Make master list of the contig numbers as a reference for mergin the data matrix values.

cmd COMMAND

```
sed -
n 1~2p ./define_core_protein_clusters/protein_clusters.fa | sed s'/>//g' | sed '1 s/^/Conti
g_ID\n/' > ./define_core_protein_clusters/master_contig_list.txt
```

Operational Protein Family Analysis

Step 25.

Merge the sample hit files to the master OPF list. After this runs, all of the resulting files should have the same number of lines because they were all merged with the same master list.

cmd COMMAND

```
mkdir ./protein_cluster_rel_abund_table/RPKM_contig_count_on_master_list
for file in $(ls ./protein_cluster_rel_abund_table/abundance_RPKM_with_only_contig_id); do
    awk 'FNR==NR {a[$1]=$2;next}{ print $1"\t"a[$1] }' ./protein_cluster_rel_abund_table/ab
undance_RPKM_with_only_contig_id/${file} ./define_core_protein_clusters/master_contig_list.
txt | sed '/[0-9]\t[0-9]/!s/$/0/' | sed '1 s/\t0//' | sed '1 s/0$//' > ./protein_cluster_re
l_abund_table/RPKM_contig_count_on_master_list/${file}
done
```

Operational Protein Family Analysis

Step 26.

Get only the abundance values so that they can be merged to the master contig list and used for distance matrix calculations.

cmd COMMAND

```
mkdir ./protein_cluster_rel_abund_table/abundance_RPKM_for_merge
for file in $(ls ./protein_cluster_rel_abund_table/RPKM_contig_count_on_master_list); do
    cut -
f 2 ./protein_cluster_rel_abund_table/RPKM_contig_count_on_master_list/$file > ./protein_cl
uster_rel_abund_table/abundance_RPKM_for_merge/${file}
done
```

Operational Protein Family Analysis

Step 27.

Merge the abundances with the OPF names.

cmd COMMAND

```
paste ./define_core_protein_clusters/master_contig_list.txt ./protein_cluster_rel_abund_tab
le/abundance_RPKM_for_merge/* > ./protein_cluster_rel_abund_table/contig_otu_table.txt
```

Operational Protein Family Analysis

Step 28.

Format the contig IDs.

cmd COMMAND

```
sed 's/|.*|\S*\t/\t/' ./protein_cluster_rel_abund_table/contig_otu_table.txt > ./protein_cl
uster_rel_abund_table/contig_otu_table_format.tsv
```

Operational Protein Family Analysis

Step 29.

Transpose the files.

cmd COMMAND

```
python transpose_tab_delim.py -
i ./protein_cluster_rel_abund_table/contig_otu_table_format.tsv -
o ./protein_cluster_rel_abund_table/contig_otu_table_transposed.txt
```

NOTES

Geoffrey Hannigan 04 Feb 2016

This relative abundance table can now be used in R for further analysis.