Profiling Soil Metagenomes Using HUMAnN2

Devin Dinwiddie

September 5, 2018

Introduction

This tutorial is meant to walk you through profiling the taxonomic and functional composition of soil metagenomic samples, as well as visualizing the results. It covers the use of HUMAnN2/Metaphlan2 (for functional and taxonomic classification), and R (for visualization and statistical evaluation) https://bitbucket.org/biobakery/humann2/wiki/Home https://www.r-project.org/. HUMAnN2 is a pipeline for profiling microbial pathways from metagenomic or metatranscriptomic sequencing data (typically short DNA/RNA reads also referred to as whole genome shotgun data). Functional profiling, aims to discover the metabolic potential and function of a particular microbial community in order to determine what the microbes of a particular community are doing IE. Metabolizing nitrogen, or breaking down cellulose. Important to note HUMAnN2 was designed to profile the human microbiome (if you didn't already guess that from the name), however with some tweaks to the commands it can be used to profile less well characterized samples such as soil microbiomes. This tutorial also assumes you are working on a UO HPC cluster where all programs needed have been installed, if this is not the case you will need to download and load all necessary programs. For a look at the standard workflow for HUMAnN2 see: https://bitbucket.org/biobakery/humann2/wiki/Home#markdown-header-main-workflow

Demo files for running HUMAnN2 have been included for working this part of the tutorial, they are Demo_S0_L001_R1_001.fastq.gz/Demo_S0_L001_R2_001.fastq.gz. These files contain a sub-sample of reads from a wine vineyard microbiome project. These are samples collected from soil and sequenced on a Ilumina Hi-Seq. Due to the small size of the samples most steps should run pretty quickly.

First step: clean your data

Typically whole genome shotgun (WGS) sequences will have lots of left over adapters and low quality sequences. Due to the nature of WGS you are getting DNA/RNA from more sources than just microbes (plants, animals, and humans) so it is important to quality filter and screen out any contaminant reads that may have come along for the ride. For the purpose of this tutorial we will go into checking for adapters and low quality sequences using fastQC and cleaning reads up using trimmomatic https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ http://www.usadellab.org/cms/?page=trimmomatic. If your sequences came from human or other animal/plant samples, or if phiX was used in the sequencing run, it is recommended that you remove samples associated with your host organism or virus

(phiX). This can be done with kneaddata https://bitbucket.org/biobakery/kneaddata/wiki/Home. For this tutorial we will assume no major contaminants.

FastQC

The following code will generate read quality reports in html format. You will need to load the required modules, make an output directory and then run fastQC on your sequence files.

```
#on talapas
#load the required modules
ml racs-eb FastQC/0.11.5-Java-1.8.0_162

#fastQC requires output directory be created befor running
mkdir $nameOfYourOutputDir # fqcOut below

#run fastQC (< 1 min to run on demo samples)
fastqc -t 28 -o fqcOut --noextract Demo_S0_L001_R* #using demo samples.
replace Undetermined_S0_L001_R* with *.gz if runnig on multiple zipped files
```

The commands used for fastQC are: (fastqc -h)

- -t --number of threads to use
- -o --path to output directory
- --noextract --Do not uncompress the output file after creating it

You should now have .html reports in your fastqc directory. Typically when dealing with WGS data you will have lots and lots of paired end files from one run. It can be a pain to look through all the fastqc reports so I recommend using multiQC http://multiqc.info/ to concatenate all of the .html files into one file. MultiQC is super easy to run and very fast.

```
#load multiQC
ml easybuild  icc/2017.1.132-GCC-6.3.0-2.27  impi/2017.1.132 MultiQC

# run multiQC from fastQC output directory (<30 sec on demo output)
multiqc .
```

This will create an output file called "multiqc_report.html". This has all of the samples reports condensed into one easy to read report.

Trimmomatic

Now that we see what the overall quality of the sequences look like, we need to clean them up before we process any further. To do this we will use trimmomatic to remove adapter content and filter reads with low quality scores. Trimmomatic looks like a mess to run but it is actually pretty easy and does a good job cleaning reads. The following code will run trimmomatic on all of the gzipped paired end files in your sample directory, it will need to

be modified if your sample file names do not look like "Demo_S0_L001_R1_001.fastq.gz, Demo_S0_L001_R2_001.fastq.gz".

```
# load trimmomatic
ml easybuild Trimmomatic/0.36-Java-1.8.0_131

# note the following message after loading *To execute Trimmomatic run: java
-jar $EBROOTTRIMMOMATIC/trimmomatic-0.36.jar*

#from the directory containing your samples
#run trimmomatic on all files using a for loop
for f in $(ls *.fastq.gz | cut -f 1-5 -d _ | sort -u | grep -v
'Undetermined')
do
java -jar $EBROOTTRIMMOMATIC/trimmomatic-0.36.jar PE ${f}_R1_001.fastq.gz
${f}_R2_001.fastq.gz \
 ${f}_R1_paired.fq.gz ${f}_R1_unpaired.fq.gz ${f}_R2_paired.fq.gz
${f}_R2_unpaired.fq.gz -threads 28
ILLUMINACLIP:$EBROOTTRIMMOMATIC/adapters/TruSeq3-PE.fa:2:30:10:1:true
LEADING:3 \
TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:42 AVGQUAL:20
done
```

The first part of the code above prepares the paired end sample names for input into trimmomatic.

- ls *.fastq.gz --list all files with .fastq.gz handle
- cut -f 1-5 -d _ --cut fields 1-5 on underscores (_) *modify if file name structure differs from demo files*
- sort -u --unique sort
- grep -v 'Undetermined' -- do not include files with "Undetermined"" in the name
- $() --place all of the above together in list for iterating over

The second part of the code runs trimmomatic on all the paired end samples. It will cut any adapter contamination it finds and filter reads that fall below an average quality score of 20. It will also cut sequence ends if below quality, and drop any reads shorter than 42 NT after trimming
https://www.urbandictionary.com/define.php?term=42

{$f} is the variable assigned from the first part of the code

- PE --paired end mode
- ${f}_R1_001.fastq.gz ${f}_R2_001.fastq.gz --input files to use
- ${f}_R1_paired.fq.gz ${f}_R1_unpaired.fq.gz --what to name the paired and unpaired output files

- -threads --number of threads to use
- ILLUMINACLIP --cut adapter and other illumina-specific sequences from the read
- Leading/Trailing -- cut bases off the start/end of a read if below a certain threshold
- SLIDINGWINDOW -- Performs a sliding window trimming approach scanning at the 5' end and clips the read once the average quality within the window falls below a threshold.
- MINLEN -- Drop the read if it is below a specified length
- AVGQUAL-- Drop the read if the average quality is below the specified level

http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf

You should now have two output files "trimmed_paired_seqs" & "trimmed_unpaired_seqs". Typically you will work with the "trimmed_paired_seqs". Also note that trimmomatic output files are not actually zipped, they just have the gz extension because that is what we said to call the output files.

Running HUMAnN2 on your sequences

Now that the reads have been trimmed and cleaned we are ready to run HUMAnN2 on our data. However, there is more housekeeping that need to be done first. For paired end reads HuMAnN2 recommends that you concatenate all reads into a single FASTA or FASTQ file. For reasons why see: https://bitbucket.org/biobakery/humann2/wiki/Home#markdown-header-humann2-and-paired-end-sequencing-data

```
# to clean things up make a new directory for the concatenated reads. It can be in the base directory for you samples or within the trimmed_paried_seqs directory ** I chose the base directory
mkdir catFiles

#from within the trimmed_paired_seqs folder
#concatenate the trimmed sample files into 1 file per paired end set.
for f in $(ls *.fastq.gz | cut -f 1-5 -d _ | sort -u | grep -v 'Undetermined')
do
cat ${f}_R1_paired.fq.gz ${f}_R2_paired.fq.gz > ../catFiles/${f}.fq.gz  #note **still not zipped **catFiles is one directory up from trimmed_paired_seqs dir.
done
```

Get the Databases

HUMAnN2 uses 2 Database sets and several utility files that do not come with the download (demos are included but not the full databases). I recommend downloading the full databases and utility files if doing anything other than test driving HUMAnN2.

```
#load the metaphlan2 module **this will atomatically load bowtie and HUMAnN2
ml metaphlan2/2
# NOTE: HUMAnN2 is loaded within the python module

#load Diamond
ml diamond/0.8.22 # make sure you load a version < 0.9.0 *later versions of
diamond do not play nice with humann2

# make a new directory in the base directory you are working in
mkdir humann2DB
```

Downlaod the Chocophlan database

HUMAnN2 uses the program metaphlan2 https://bitbucket.org/biobakery/metaphlan2 in order to assign taxonomy to your metagenome sequences. Metaphlan2 works by aligning reads using bowtie2 http://bowtie-bio.sourceforge.net/bowtie2/index.shtml to the chocophlan database which is a pangenome database built by clustering coding sequencing from NCBI genomes. Below is code to download the chocophlan database.

```
# download the  chocoplan database
humann2_databases --download chocophlan full ./humann2DB --update_config no
#Note: if on Talapas do not update the config file as this will change global
database directory to your folder and not everyone who may want to use
HUMAnN2 has access.
```

Download *one* of the following Uniref databases:

- UniRef90 EC filtered
- UniRef90
- UniRef50 EC filtered
- UniRef50 database

HUMAnN2 uses Diamond https://github.com/bbuchfink/diamond to align reads to a protein reference database. There are several Uniref protein alignment databases to choose from. UniRef90 is a good default choice when profiling well characterized systems, since it is comprehensive and non-redundant. UniRef50 clusters can be very broad, so there's a risk that the cluster representative might not reflect the function of the homologous sequence(s) found in your data set. One situation where UniRef50 might be preferable is when dealing with *very poorly characterized microbiomes*. In that case, requiring reads to map at 90% identity to UniRef90 might be too stringent, and so mapping at 50% identity to UniRef50 could explain a larger portion of sample reads *with reduced resolution however.

Since this tutorial is working with soil samples and they are not well characterized we will work with the uniref50 database

```
# downlaod the uniref DB
humann2_databases --download uniref uniref50_diamond ./humann2DB --
update_config no
```

Download the utility mapping files

The output from HUMAnN2 has RPK counts with uniref names attached. In order to analyse the results it is often useful to have kegg ID's or EC names attached to the data. HUMAnN2 can do this but it needs a set of utility mapping files to accomplish.

```
humann2_databases --download utility_mapping full ./humann2DB --update_config
no
```

Running HUMAnN2

Now that the housekeeping stuff is out of the way we are ready to run our sequences through the HUMAnN2 pipeline. The following code will run all of your concatenated files through the pipeline. This can take several days (9+) to run on large data sets (40+ samples), so be aware and set the wall-clock limits in your script accordingly.
https://hpcrcf.atlassian.net/wiki/spaces/TCP/pages/7312376/Quick+Start+Guide

```
#from base directory
#run humann2
for f in $(ls ./catFiles/*.fq.gz)
do
humann2 --memory-use maximum --threads 28 --metaphlan-options="--mpa_pkl
/packages/metaphlan2/2/bin/metaphlan_databases/mpa_v20_m200.pkl --bowtie2db
/packages/metaphlan2/2/bin/metaphlan_databases --bt2_ps very-sensitive-local"
--nucleotide-database ./humann2DB/chocophlan/ --translated-query-coverage-
threshold 0.6 --protein-database ./humann2DB/uniref50/ --input ${f} --output
humann2_50_out/
done
```

The commands used for HUMAnN2 are: (humann2 -h)

- --memory-use maximum --how much memory to use
- --threads --number of threads to use
- --metaphlan-options="" --the options for metaphlan2 *note the format* this is needed in order to get metaphlan to function on Talapas. It directs to the bowtie DB and the metaphlan .pkl file
- --nucleotide-database --location of the chocophlan DB
- --translated-query-coverage-threshold --query coverage threshold for translated protein alignments
- --protein-database -- location of the uniref DB
- --input/output --location of input files and where you want output files kept.

A couple of things to note: we used "--bt2_ps very-sensitive-local" to do a sensitive local alignment with bowtie. This should help get better alignments when using longer read lengths (>100 NT). We also used a 60% protein alignment threshold (instead of the default 90%) which determines how much of the read must align to a protein. This was also to loosen the profiling requirements due to soil microbiomes being not well classified. Feel free to mess around with the alignment threshold and alignment type to suit your own data.

Working with HUMAnN2 output files

The files output from HUMAnN2 should be in your output folder humann2_50_out if your following this tutorial. In this folder you will see several files. For each sample run you should have a pathabundace.tsv, genefamilies.tsv and a pathcoverage.tsv file, as well as a temp folder.

- genefamiles.tsv --This file details the abundance of each gene family in the community. Gene families are groups of evolutionarily-related protein-coding sequences that often perform similar functions.
  - Gene family abundance is reported in RPK (reads per kilobase) units to normalize for gene length; RPK units reflect **relative** gene (or transcript) copy number in the community
- pathabundace.tsv --This file details the abundance of each pathway in the community as a function of the abundances of the pathway's component reactions, with each reaction's abundance computed as the sum over abundances of genes catalyzing the reaction.
- pathcoverage.tsv --Pathway coverage provides an alternative description of the presence (1) and absence (0) of pathways in a community, independent of their quantitative abundance.
- temp output folder --This folder contains all the bowtie and diamond alignment output files as well as the metaphlan bugslist.tsv which a list of microbes found in the samples (can be useful, however note the genefamiles has bug names attached to each gene)

For more information on these files see:
https://bitbucket.org/biobakery/humann2/wiki/Home#markdown-header-output-files

Important note: because the genefamilies files are relative gene copy number and not raw counts these output files are not suitable for differential expression analysis with programs such as DESeq2 or edgeR. https://support.bioconductor.org/p/104414/

Getting HUMAnN2 output ready for R

In order to work with the output files in a program like R it is recommended the counts be normalized. The options for normalization are relative abundance (relab) or counts per million (Cpm). Either method will work, for this tutorial we will go with relative abundance. Before exporting to R I also like to split the data to get abundance by gene, regardless of taxonomic sub-match, and convert the Uniref ID/names to Kegg ID/names.

Group all tables together

```
#from within the directory humann2_50_out
humann2_join_tables -i ./ --file_name genefamilies -o genefamilies_u50.tsv
#This will group all the genefamiles tables generated into one table
```

Split the full table into stratified and unstratified tables

stratified table contains each gene with microbe name so several of the genes are repeated if they come from different bugs, unstratified puts counts for genes together regardless of which bug they belong too

```
humann2_split_stratified_table -i genefamilies_u50.tsv -o ./ # this will
generate two tables stratified and unstratified.
```

Re-Group unstratified table to kegg ID

```
humann2_regroup_table -i genefamilies_u50_unstratified.tsv -c
../humann2DB/utility_mapping/map_ko_uniref50.txt.gz --output ko_u50_RPKs.txt
```

Remove ungrouped and unmapped counts

```
cat ko_u50_RPKs.txt | grep -v 'UNMAPPED' |grep -v 'UNGROUPED'
>ko_u50_RPKS_clean.txt
```

Ungrouped and unmapped reads typically make up a substantial percentage of the counts. Feel free to leave them in if you want.

Re-name the table with kegg names

```
humann2_rename_table -i ko_u50_RPKS_clean.txt -c
../humann2DB/utility_mapping/map_ko_name.txt.gz -o ko_u50_renamed.tsv # this
will attach the gene names to the kegg ID column
```

normalize counts to relative abundance

```
humann2_renorm_table -i ko_u50_renamed.tsv --units relab -o
ko_u50_relabun.txt # change --units cpm to get counts per million
```

Using R to make HeatMaps of certain gene family abundances

The following is a guide to making a heat map with gene family data output from HUMAnN2. The data come from a subset of samples collected from wine vineyard soils in the Pacific Northwest. The associated files include the gene family relative abundance data, as well as the metadata associated with the samples, and a map to the Kegg pathway descriptions. These data are intended for learning only and have been altered to protect the real data.

The metadata file contains the sample name, whether samples were taken from directly below a vine or in the alley between two vine rows, as well as the management style of the vineyard (conventional or organic).

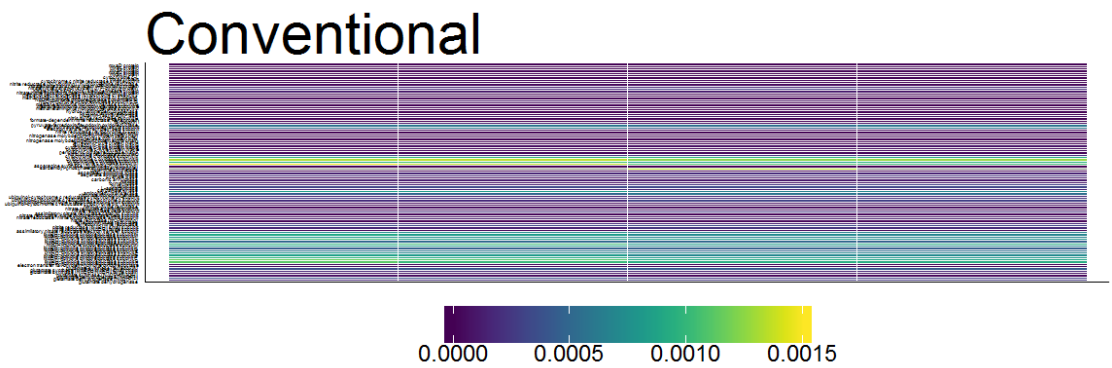
For this tutorial we will focus on the nitrogen metabolism pathway across management styles. Demo files have been included for this part of the tutorial. They are a gene family counts table -ko_u50_Demo_relabun.txt, a metadata file -Demo_mapping.txt, and a Kegg ID/pathway map -ko_map.txt. You will need to make sure these files are in the path of your working directory in order to run correctly.

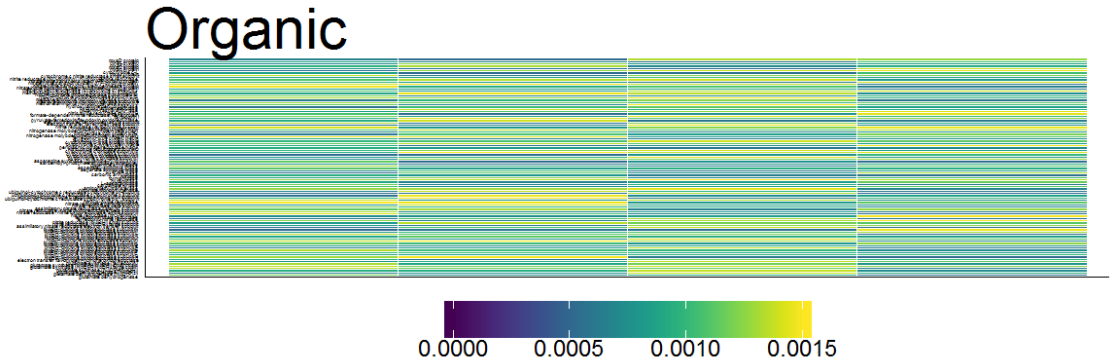
Heatmap

```
#load the librarys
library(reshape2)
library(ggplot2)
library(viridis)
library(plyr)
library(ggpubr)

#read in the data
gene_counts=read.delim("ko_u50_Demo_relabun.txt",header = T) # read in the
relab gene count data
meta=read.delim("Demo_mapping.txt") # read in the meta data
Ko_map=read.delim("ko_map.txt",header=T) # read in the kegg pathway map


# organize the data
colnames(gene_counts)=gsub("_S.*","",colnames(gene_counts)) #rename the
headers to match meta data sample names *replace _S and everything after with
a space

colnames(Ko_map) <- c("Level_1", "Level_2", "Level_3", "ko", "Description",
"ID", "EC") # rename headers of kegg map

colnames(meta)[1]='ID' # rename the first column of the meta data to ID for
merging later

gene_counts$ID=sub(":.*","",gene_counts$X..Gene.Family) # make a ID column of
ko names to match kegg map file **for merge later


# grab the kegg data associated with Energy Metabolism --specificaly Nitrogen
metabolism
koNit <- subset(Ko_map, Level_2 == "Energy Metabolism" & Level_3=='Nitrogen
metabolism') # look at just energy metabolism --nitrogne metabolism pathways

#merge the data together
full_tab=merge(gene_counts,koNit,by='ID')  # merge gene abundance table with
kegg map

long_tab=melt(full_tab) #convert to long format

# make a column of gene names
```

```
long_tab$Gf=gsub("\\[.*","",long_tab$X..Gene.Family)
long_tab$Gf=gsub(".*:","",long_tab$Gf)

#get rid of the columns not needed
long_tab1=long_tab[,9:11]

colnames(long_tab1)[1]="ID" #change column name to merge with meta data


final_tab=merge(long_tab1,meta,by="ID") # merge meta data with long format
data

final_tab=subset(final_tab,Gf!=" NO_NAME") # some of the gene names are
unwanted * remove those


# we are going to create 2 heatmaps for each management type then put them
together in 1 figure
# first we need to subset the data by management type

conv=subset(final_tab,final_tab$management=='conv') #subset conventional
management
org=subset(final_tab,final_tab$management=='org') #subset organic management

# set the levels for plotting
conv$Gf=factor(conv$Gf,levels=unique(conv$Gf))
org$Gf=factor(org$Gf,levels=unique(org$Gf))

#length(unique(conv$Gf))#97 unique gene names
#length(unique(org$Gf)) #97 as well


#make some heatmaps with ggplot

#conventional
gc <- ggplot(conv, aes(x=ID, y=Gf, fill=value)) +
  geom_tile(color="white", size=0.1)+
  scale_fill_viridis(name="scale",limits=c(0,0.0015))+ # I already know the
limits to set remove limits=() if running your own data
  labs(x=NULL, y=NULL, title="Conventional")+
 theme(axis.ticks=element_blank())+
  theme(axis.text.y=element_text(color=c(rep('black',96))))+
 theme(legend.position="bottom")+
 theme(legend.key.size=unit(1, "cm"))+
 theme(legend.key.width=unit(2, "cm"))+
  theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
panel.background = element_blank(), axis.line = element_line(colour =
"black"),strip.text = element_text(colour = 'white'),text =
```

```
element_text(size=5),legend.text=element_text(size=18),axis.text.x =
element_blank(),axis.ticks.x = element_blank(),legend.title =
element_blank(),plot.title = element_text(size=44))+theme(strip.text =
element_text(size = 32))

#ORGANIC
go <- ggplot(org, aes(x=ID, y=Gf, fill=value)) +
  geom_tile(color="white", size=0.1)+
  scale_fill_viridis(name="scale",limits=c(0,0.0015))+ # I already know the
limits to set remove limits=() if running your own data
  labs(x=NULL, y=NULL, title="Organic")+
 theme(axis.ticks=element_blank())+
  theme(axis.text.y=element_text(color=c(rep('black',96))))+
 theme(legend.position="bottom")+
 theme(legend.key.size=unit(1, "cm"))+
 theme(legend.key.width=unit(2, "cm"))+
  theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
panel.background = element_blank(), axis.line = element_line(colour =
"black"),strip.text = element_text(colour = 'white'),text =
element_text(size=5),legend.text=element_text(size=18),axis.text.x =
element_blank(),axis.ticks.x = element_blank(),legend.title =
element_blank(),plot.title = element_text(size=44))+theme(strip.text =
element_text(size = 32))


# put the 2 figures into 1 figure for presentaion
fin=ggarrange(gc,go,
          ncol = 1, nrow =2)

fin
```






```
#let's check real quick if there is a significant difference in nitrogen
metabolism genes across management types

summary(aov(value~management,data=final_tab))

##              Df    Sum Sq   Mean Sq F value Pr(>F)
## management    1 1.032e-04 1.032e-04   971.9 <2e-16 ***
## Residuals   782 8.299e-05 1.100e-07
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There are many more analyses that can be done with the data generated from HUMAnN2. Try looking at the stratified gene family tables to see which bugs were associated with Nitrogen Metabolism, or some other pathway. The beauty and pain of metagenomic data is the analysis possibilities are endless. So good luck and have fun.