Analysis of tadpole skin microbiome data using oligotyping

Will Shoemaker

August 20, 2015

This document goes through the steps taken to analyize tadpole microbiome data using the oligotyping method. What oligotyping does is to look at the sequences that are clustered within an OTU and how they differ at the level of individual nucleotides. It uses information theory to look at sites along aligned 16s rRNA sequence data with a high level of entropy. A high level of entropy at a site means that there's a high level of uncertainty, so you have nucleotide diversity.

First, I had to reformat the FASTA file of the aligned, p-filtered sequences for the oligotyping program to run. This is a really annoying thing about oligotyping, the pipeline has these formatting standards that aren't found in QIIME or mothur what the formatting means isn't explained well on the webpage. The example is pasted below

```
>Sample-01_ReadX
GTTGAAAAAGTTAGTGGTGAAATCCCAGA
>Sample-01_ReadY
GTTGAAAAAGTTAGTGGTGAAATCCCAGA
>Sample-01_ReadZ
GGTGAAAAAGTTAGTGGTGAAATCCCAGA
>Sample-02_ReadN
GTTGAAAAAGTTAGTGGTGAAATCCCAGA
>Sample-02_ReadM
GTTGAAAAAGTTAGTGGTGAAATCCCAGA
```

After some fooling around, the spacing and the characters don't matter. If you're running oligotyping, all that matters is that your sample information is contained in **exactly** eight characters at the start of the FASTA header.

Because my R skills are not as strong as my Python skills, I wrote this in Python. I've copied and pasted the Python code into the block below. It's also written in a standard .py file.

```
fasta = open('/Users/WRShoemaker/Desktop/Harris Lab/Tadpole_Sequences/'
             'OTUTable_Tadpoles_97uclust/pynast_aligned_seqs/'
             'extractedseqs250_rep_set_aligned_pfiltered.fasta')
newfasta = open('/Users/WRShoemaker/Desktop/Harris Lab/Tadpole_Sequences/'
                'OTUTable_Tadpoles_97uclust/'
                'extractedseqs250_rep_set_aligned_pfiltered_Reformated.fasta', 'w')
def fasta_parse(fasta):
  for line in fasta:
        if line.startswith('>'):
            split = line.split()
            OTU = split[0].split('>')
            sample = split[1][0:4]
            tadpole, treatment = sample[:len(sample)/2], sample[len(sample)/2:]
            read = split[1][5:]
            newname = ">" + treatment + "T" + "_" + "00" + tadpole + "_" + \
  OTU[1] + "_" + read + "\n"
            newfasta.write(newname)
```

```
        else:
            pass
            newfasta.write(line)
fasta_parse(fasta)
fasta.close()
newfasta.close()
```

We now have our correctly formatted file. Next we set our R directory.

```
rm(list=ls())
getwd()
```

```
## [1] "/Users/WRShoemaker/github/HorizEnvTransfer"
```

```
#setwd('~/Desktop/Harris Lab/Tadpole_Sequences/OTUTable_Tadpoles_97uclust/')
getwd()
```

```
## [1] "/Users/WRShoemaker/github/HorizEnvTransfer"
```

For now we need two packages. Phyloseq and otu2ot. Phyloseq is a 16s data management package with a lot of dependencies. It's useful if you've got something like a BIOM formatted table from QIIME and you want to get a site-by-species matrix. Otu2ot is a wrapper for the oligotyping pipeline

```
## Loading required package: phyloseq
## Creating a generic function for 'nchar' from package 'base' in package 'S4Vectors'
## Loading required package: otu2ot
```

Next we import our files using phyloseq.

```
otufile <- "~/Desktop/Harris Lab/Tadpole_Sequences/OTUTable_Tadpoles_97uclust/OTUTable.biom"
mapfile <- "~/Desktop/Harris Lab/Tadpole_Sequences/OTUTable_Tadpoles_97uclust/Mapping_Tadpoles.txt"
trefile <- "~/Desktop/Harris Lab/Tadpole_Sequences/OTUTable_Tadpoles_97uclust/rep_set.tre"
envir <- import_qiime_sample_data(mapfile)
myData <- import_biom(otufile, trefile)
myData <- merge_phyloseq(myData,envir)
```

Let's look at our taxonomic ranks

```
rank_names(myData)
```

```
## [1] "Rank1" "Rank2" "Rank3" "Rank4" "Rank5" "Rank6" "Rank7"
```

The taxonomic rank labels are incorrect in the file, so we can change that.

```
colnames(tax_table(myData)) <- c(k = "Kingdom", p = "Phylum", c = "Class",
                                 o = "Order", f = "Family", g = "Genus", s = "Species")
rank_names(myData)
```

```
## [1] "Kingdom" "Phylum"  "Class"   "Order"   "Family"  "Genus"   "Species"
```

This makes data manipulation easier.

Phyloseq merges all the relevant data into a single object, so we can get useful information from our samples. For example, here we are getting the number of OTUs across all samples and what sample variables we have

```
ntaxa(myData)
```

```
## [1] 128365
```

```
sample_variables(myData)
```

```
##  [1] "X.SampleID"           "BarcodeSequence"      "LinkerPrimerSequence"
##  [4] "Day"                  "Barcode"              "Tadpole"
##  [7] "Inoculated"           "Tank"                 "Treatment0"
## [10] "Treatment1"           "Treatment2"           "Description"
```

Next we extract the names of those OTUs.

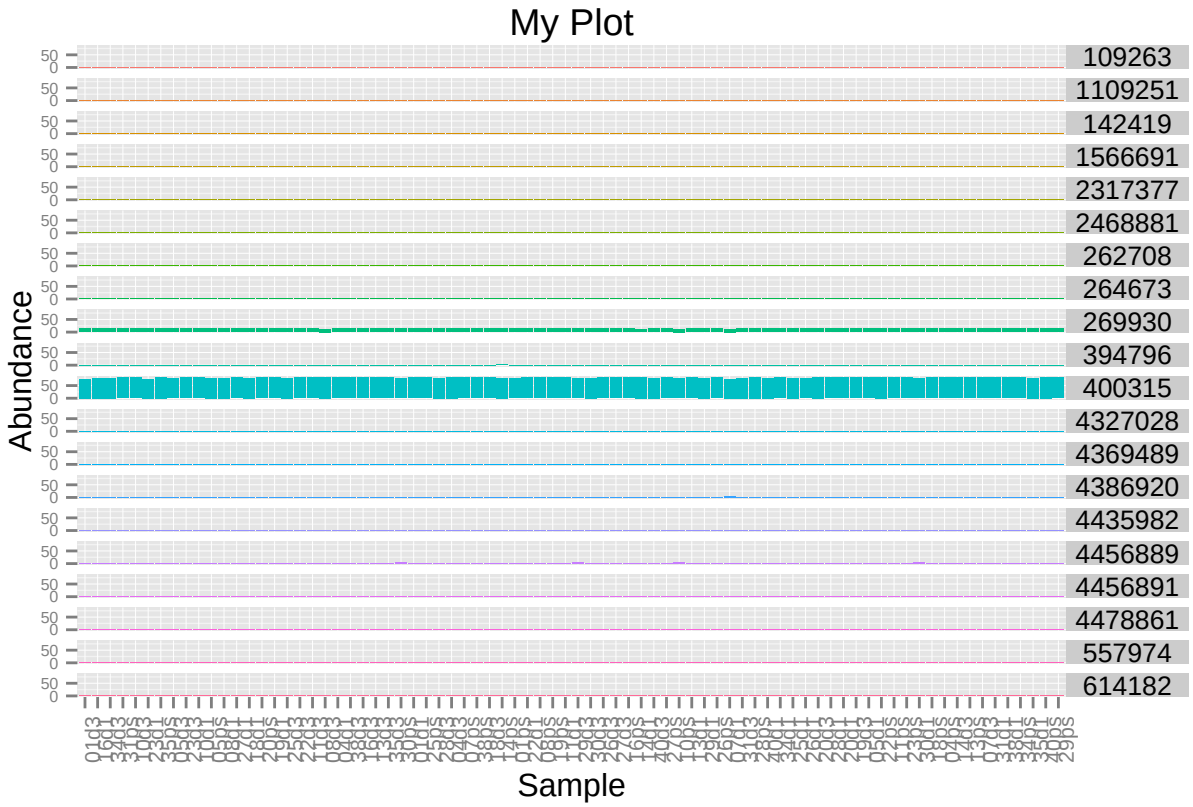
```
GP.chl <- subset_taxa(myData, Genus == "g__Pseudomonas")
```

From Eria's graphs we know that *Pseudomonas* is disproportianatelly dominant across all samples. Let's look at the OTUs that have been assigned the genus label *Pseudomonas*. First as a plot of OTU abundances labelled as *Pseudomonas* across samples.

```
library("ggplot2")
library("stringr")
library("reshape2")
library("grid")

Pseudo.SbyS <- otu_table(GP.chl)
s_num<-dim(Pseudo.SbyS)[2]
totals<-colSums(Pseudo.SbyS)

s_num<-dim(Pseudo.SbyS)[2]
otu.perc<-matrix(rep("NA",times=(dim(Pseudo.SbyS)[1]*s_num)),nrow=dim(Pseudo.SbyS)[1],ncol=s_num)
rownames(otu.perc)<-rownames(Pseudo.SbyS)
colnames(otu.perc)=colnames(Pseudo.SbyS)
totals<-colSums(Pseudo.SbyS)

top.otus = as.data.frame(head(sort(rowSums(Pseudo.SbyS), decreasing = T),20))

for(s in c(1:s_num)){
  vec<-(Pseudo.SbyS[,s]/totals[s])*100
  otu.perc[,s]<-vec
  rm(vec)
}

write.table(otu.perc, "otu_perc.txt", col.names = NA, row.names = T, sep = "\t")
otu.perc = read.table("otu_perc.txt", sep = "\t", row.names=1, header=T, check.names=F)

otu.20.perc = otu.perc[rownames(top.otus),]
otu.20.melt.perc = melt(cbind(rownames(otu.20.perc),otu.20.perc))
```

```
## Using rownames(otu.20.perc) as id variables
```

```
colnames(otu.20.melt.perc)[1] = "OTU"
colnames(otu.20.melt.perc)[2] = "Sample"
colnames(otu.20.melt.perc)[3] = "Abundance"

#pdf("my_plot.pdf", height = 8, width = 8)
otu.plot.data.perc = ggplot(otu.20.melt.perc, aes(x=Sample, y = Abundance, fill = OTU))
otu.barplot.perc = otu.plot.data.perc + geom_bar(stat = "identity", position = "stack")
otu.barplot.perc + facet_grid(OTU ~.) + ggtitle("My Plot") +
  theme(legend.position = "null", strip.text.y = element_text(size = 10, angle = 0),
        axis.text.x = element_text(size = 8 , angle = 90), axis.text.y = element_text(size = 6),
        panel.grid.major = element_line(size = 0.1)) + scale_y_continuous(breaks = c(0,50))
```




```
#dev.off()
```

The x-axis contains samples from individual tadpoles. The y-axis contains *Pseudomonas* OTUs

We want to select the 16s rRNA reads from our reformatted FASTA file that correspons to an OTU that has been given *Pseudomonas* as a taxonomic assignment. To do this we first need the names of the OTUs belonging to *Pseudomonas*

```
Pseudo.names <- taxa_names(GP.chl)
```

Then we import our FASTA file. We're using the package seqinr. It has a function that reads FASTA files.

```
library("seqinr")
```

```
## Loading required package: ade4
```

```
fastafile<- read.fasta(file = "~/Desktop/Harris Lab/Tadpole_Sequences/OTUTable_Tadpoles_97uclust/extract
                       seqtype = "DNA",as.string = TRUE, set.attributes = FALSE)
```

Let's look at the first few lines to make sure that our file was imported.

```
head(fastafile)
```

```
## $d3T_0014_10001_3902899
## [1] "----------------------------------------------------tacggagggtgcaagcgttaatcggaattactgggcgtaaagcg
##
## $d1T_0035_1000876_814427
## [1] "----------------------------------------------------tacgtagggtgcgagcgttgtccggaattattgggcgtaaaggg
##
## $d3T_0010_1001007_2189641
## [1] "----------------------------------------------------tacagagggtggcaagcgttatccggaattattgggcgtaaagcg
##
## $d3T_0013_1001564_3865740
## [1] "----------------------------------------------------tacgtaggggcaagcgttgtccggaattattgggcgtaaagcg
##
## $d1T_0026_100208_5702854
## [1] "----------------------------------------------------tacgtagggtgcaagcgttaatcggaattactgggcgtaaagcg
##
## $d1T_0038_1002759_2794214
## [1] "----------------------------------------------------tacagagg-tgccagcgttaggcggaatcactgggcttaaagcg
```

Okay, so next we need to pull out all sequences with a FASTA header that corresponds to an OTU label with *Pseudomonas* as a taxonomic assignment.

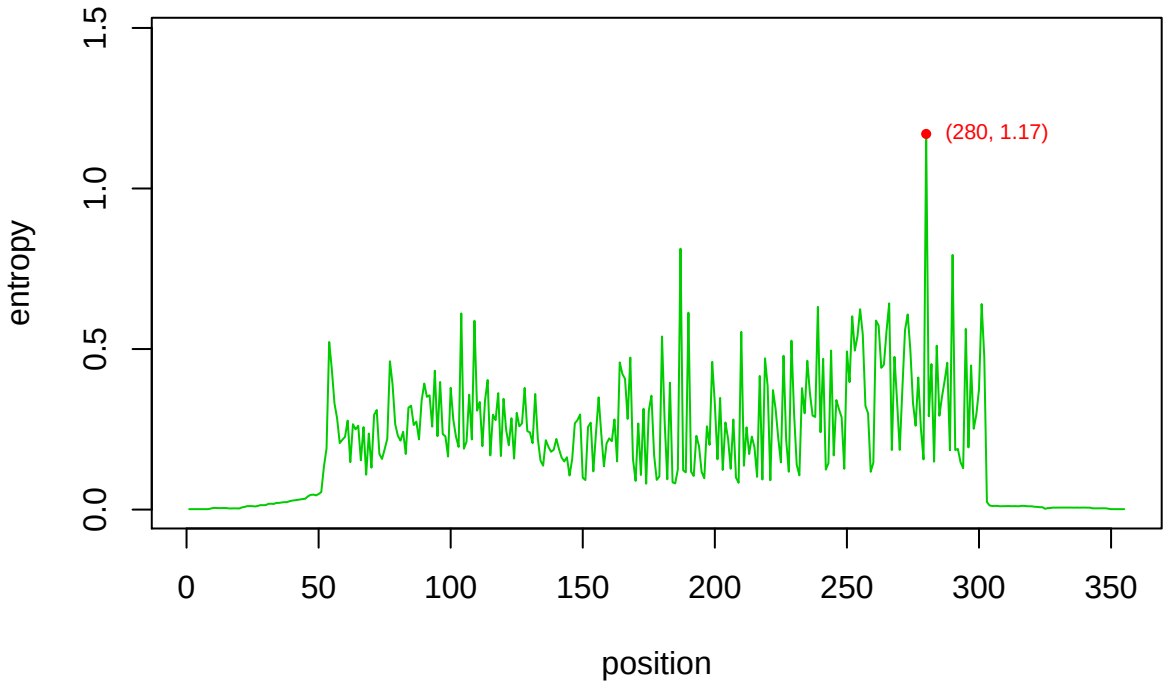
```
test <- fastafile[lapply(strsplit(names(fastafile), "_"), `[[`, 3) %in% Pseudo.names]
```

Like I said, oligotyping is weird. We need to export our reads into a new FASTA and re-import them for oligotyping to work.

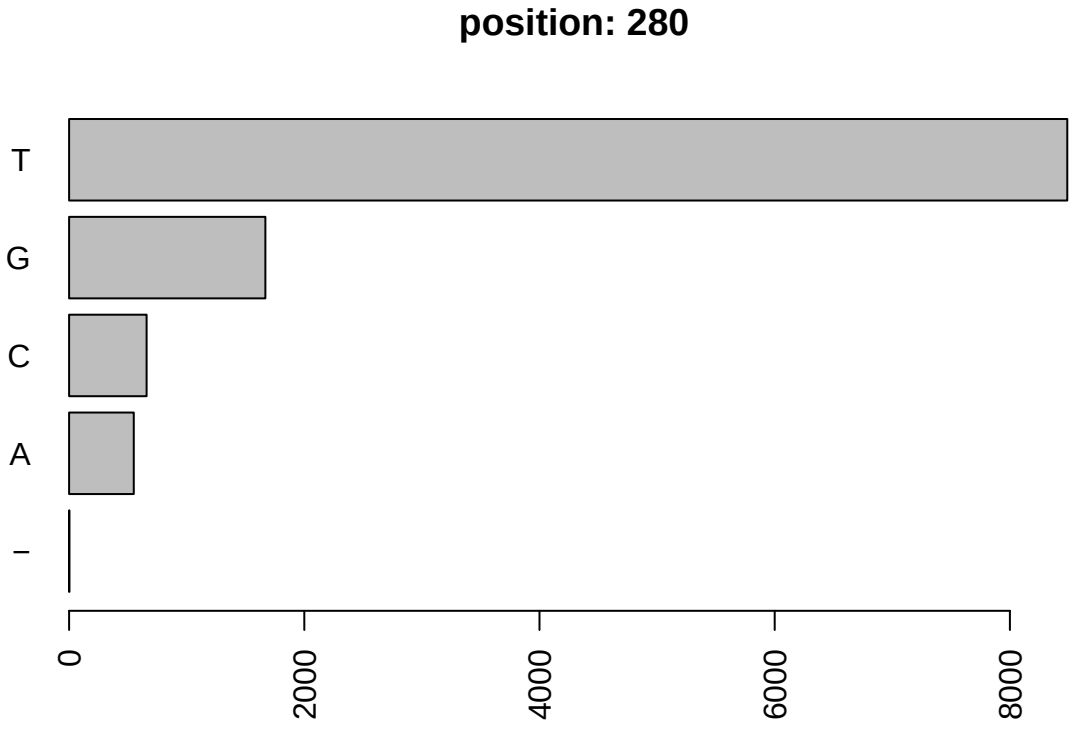
```
write.fasta(sequences = test, nbchar = 60, names = names(fastafile),
            file.out = "./extractedseqs250_rep_set_aligned_pfiltered_Pseudo_only.fasta", open = "w")
File <- "./extractedseqs250_rep_set_aligned_pfiltered_Pseudo_only.fasta"
```

We then analyze the entropy at each site along the aligned 16s sequences

```
Pseudo.OT <- MED(File, minseq = 21, entropymin = 0.6, Plot = T)
```




```
##
## Position: 280
##         -      A      C      G       T
## Nber 2.00 550.00 659.00 1669.0 8488.00
## Prop 0.01   1.55   1.86    4.7   23.91
```



```
head(Pseudo.OT)
```

```
##        1        2        3        4        5        6
##   "CATG" "TTTTAC" "TTTTAC" "TTTTAC" "TTTTAC" "TTTTAC"
```

Below we're checking if our environmental label in the FASTA header is correct.

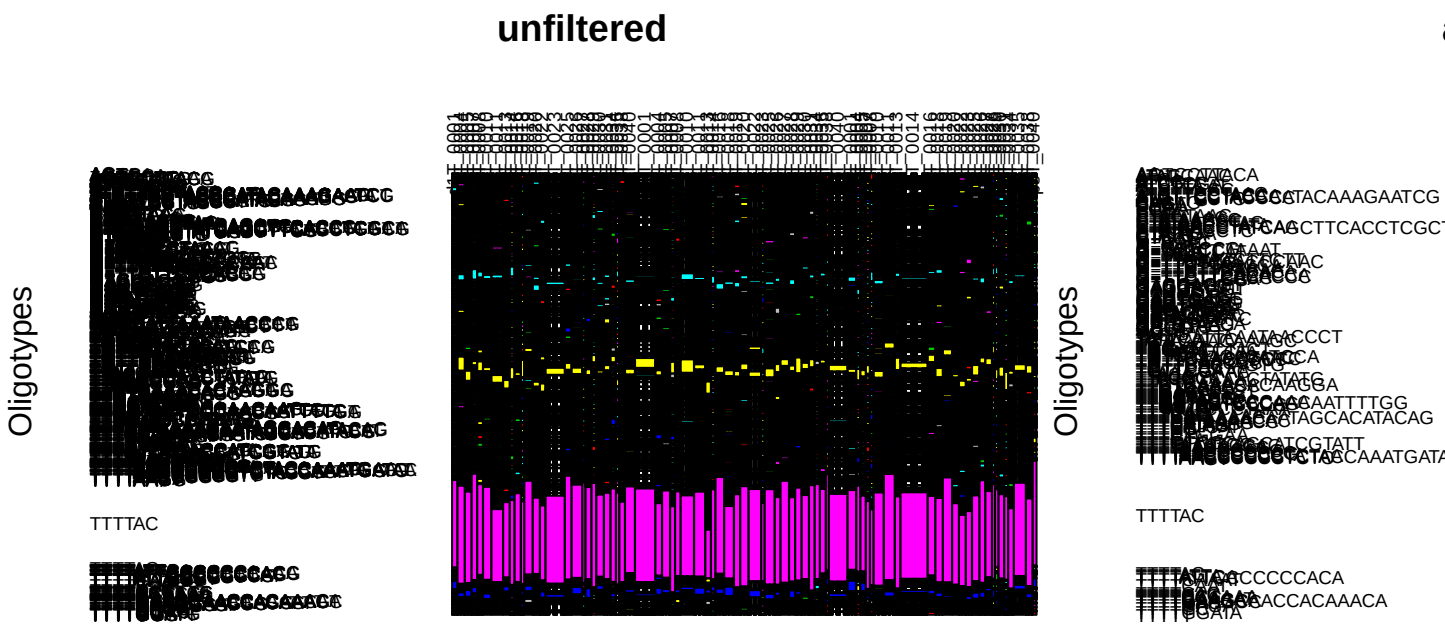
```
GetEnvironmentDatafromFileR(File, Start = 2, Stop = 9, test = T)
```

```
## [1] "d3T_0014"
```

```
ENV <- GetEnvironmentDatafromFileR(File, Start = 2, Stop =9, test = F)
```

So we filter our oligotypes at an abundance of 10.

```
Table10 <- SampleXOT_Table(
  Pseudo.OT,
  ENV = ENV,
  mosaicPlot = T,
  filterByMinAbund = 10
)
```



So there's a single oligotype that is abundant across samples.

However, after learning how to run the oligotyping pipeline in Python, translating that into R, and reworking my code that uses mothur files for QIIME files, I realized that I forgot something very basic. There's no reason to run oligotyping if the OTU that you're interested in looking at contains only one sequence.

Let's see how many sequences belong to our most abundant OTU.

```
read.count <- fastafile[lapply(strsplit(names(fastafile), "_"), `[[`, 3) == '400315']
read.count
```

```
## $d3T_0018_400315_4
## [1] "----------------------------------------------------tacagagggtgcaagcgttaatcggaattactgggcgtaaagcg
```

```
length(read.count)
```

```
## [1] 1
```

There's only one, so all that work, while correct and usable for analyzing other datasets, doesn't do a thing here.

While we're here let's blast the sequence and look at what OTU 400315 matches.

We can do this with the Bioconductor package "annotate"

```
#library('annotate')
#Pseudo.blast <- blastSequences(x = read.count, hitListSize = 10, timeout=50, as="data.frame")
#head(Pseudo.blast)
```

This package isn't playing well with RMarkdown, so I'm printing the output below. The package works fine in basic R, but not in RMarkdown.

```
 Hit_num                      Hit_id
1       1 gi|913150254|gb|KR085915.1|
2       2 gi|913150102|gb|KR085817.1|
3       3 gi|834960580|gb|KP792401.1|
4       4 gi|834960578|gb|KP792400.1|
5       5 gi|831250985|gb|KP406426.1|
6       6 gi|831250968|gb|KP406409.1|
                                                                                Hit_def
1                         Pseudomonas gessardii strain IHBB 9846 16S ribosomal RNA gene
2                         Pseudomonas gessardii strain IHBB 9179 16S ribosomal RNA gene
3                                      Pseudomonas sp. PO366 16S ribosomal RNA gene
4                                      Pseudomonas sp. PO283 16S ribosomal RNA gene
5 Uncultured Pseudomonas sp. clone C10-G1409301393-3-12-M13F(-47) 16S ribosomal RNA gene
6  Uncultured Pseudomonas sp. clone B03-G1407160532-2-8-M13F(-47) 16S ribosomal RNA gene
  Hit_accession Hit_len Hsp_num Hsp_bit-score Hsp_score   Hsp_evalue
1      KR085915    1484       1       452.129       500 1.69425e-123
2      KR085817    1504       1       452.129       500 1.69425e-123
3      KP792401    1436       1       452.129       500 1.69425e-123
4      KP792400    1477       1       452.129       500 1.69425e-123
5      KP406426    1071       1       452.129       500 1.69425e-123
6      KP406409    1162       1       452.129       500 1.69425e-123
```

And our top hit is *Pseudomonas gessardii*, a fluorescent Gram-negative bacterium that was isolated from natural mineral water in France.