Metagenomic Analysis

Authors: Tony Kabilan Okeke, Ifeanyi Osuchukwu
Date: 02.28.2022

Identification of the types and abundances of the micobial species living on the human body is important for understanding the interactions of these microorganisms among themselves and with the host. A change or difference in the composition of the microbiome has been implicated in a number of diseases.

In this assignment, you will analyze a (hypothetical) sample from oral cavity and identify the species and abundances of the bacteria. You are given the short reads obtained from Next-Gen sequencing of the sample. The data used for this assignment was extracted/adapted from the publication *"Next Generation Sequencing Data of a Defined Microbial Mock Community"*

The data used here has been manipulated and should not be used to draw any scientific conclusions.

To simplify the assignment, you are only asked to find out the fraction of 3 microorganism species.

In [ ]:
```
%load_ext autoreload
```

In [ ]:
```
%autoreload 2
# Imports
import matplotlib.pyplot as plt
import pandas as pd
import simplesam
import bmes
import re
import os
from Bio import SeqIO

%matplotlib inline
```

In [ ]:
```
# Set up datadir and BWA path
DATADIR = bmes.datadir() + '/hwmetagenome/'
if not bmes.isfolder(DATADIR):
    bmes.mkdirif(DATADIR)

if bmes.computername() == 'KabilansPC':
    BWAEXE = "/home/kabil/.anaconda3/envs/blast/bin/bwa"
else:
    BWAEXE = bmes.bwaexe(); # Ahmet's installation
```

In [ ]:
```
# Definitions
def download_genome(accesion: str, file: str) -> str:
    if not bmes.isfileandnotempty(DATADIR + file):
        bmes.downloadurl(
            f"https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id={accesion}&rettype=fasta",
            DATADIR + file
        )
    return (DATADIR + file)
```

Download Next-Gen Sequence Data (fastq file)

A small data file is provided so you can develop and test your analysis code quickly. Your final results should be based on the largest data file.

In [ ]:
```
#fastqfile = "http://sacan.biomed.drexel.edu/ftp/SRR3656745_pass.randsample.select.102.fastq"

# Uncomment the following lines to test your code on the larger file,
# once it works for the smaller file above.
# fastqfile = "http://sacan.biomed.drexel.edu/ftp/SRR3656745_pass.randsample.select.1001.fastq"

# Uncomment the following line to test your code on this larger file once it
# works for the smaller file(s) above.
#fastqfile = "http://sacan.biomed.drexel.edu/ftp/SRR3656745_pass.randsample.select.10001.fastq"

# Uncomment the following line to test your code on this larger file, once it
# works for the smaller file(s) above.
# Your final results should be based on this data file.
fastqfile = "http://sacan.biomed.drexel.edu/ftp/SRR3656745_pass.randsample.select.86588.fastq"
```

```
if bool(re.search('^(https?://|ftps?://)', fastqfile)):
    fastqfile = bmes.downloadurl(fastqfile, DATADIR + "randsample.86588.fastq")
```

Download the Genomes

Download the following genomes:

- Olsenella uli (NC_014363.1)
- Segniliparus rotundu (NC_014168.1)
- Escherichia coli K-12 (NC_000913.1)

In [ ]:
```
genomes = {
    "O_uli": download_genome("NC_014363.1", "Olsenella_uli.fasta"),
    "S_rotundu": download_genome("NC_014168.1", "Segniliparus_rotundu.fasta"),
    "E_coli": download_genome("NC_000913.1", "Escherichia_coli.fasta")
}
```

Basic Mapping: Map the `FASTQ` file to the Following Three Genomes Separately

Use `BWA` to map the reads to each of the genomes.

Check the result of the BWA to assign each short read to one of the genomes, or assign it "Unknown" if it cannot be mapped to any of the genomes.

- If a short read does not map to any organism, increase the count for the Unknown group by 1.
  Not-aligned short reads have a ReferenceName `'*'` and a Position 0.
- If a short read maps to only one organism, increase the count for that organism by 1.
- If a short read maps to more than one organism, use a fractional increase in the organism counts.
  E.g., if a read is mapped to the first and third organisms only, increase the counts for the first and third organisms by 0.5, and do not increase the count for the second organism.
  E.g., if a read is mapped to all three organisms, increase each of their counts by 1/3.

Note: `BWA` may return more than one hit within a genome for a read. You need to make sure such hits are not counted multiple times for a genome.

In [ ]:
```
# Index the reference genomes
for species in genomes:
    if not bmes.isfileandnotempty(genomes[species] + '.bwt'):
        cmd = f"{BWAEXE} index '{genomes[species]}'"
        os.system(cmd)
    else:
        print(f"{species} genome already exists.")
```

```
O_uli genome already exists.
S_rotundu genome already exists.
E_coli genome already exists.
```

In [ ]:
```
# Align sequence to reference genomes
samfiles = dict()

for species in genomes:
    samfiles[species] = f"{DATADIR}{fastqfile.split('/')[-1]}_{genomes[species].split('/')[-1]}.sam"
    if not bmes.isfileandnotempty(samfiles[species]):
        cmd = f"{BWAEXE} mem '{genomes[species]}' '{fastqfile}'"
        bmes.system_redirecttofile(cmd, samfiles[species])
```

In [ ]:
```
# Parse sam files into data frames
sams = {k:[] for k in genomes}

for species in genomes:
    with open(samfiles[species], 'r') as f:
        samiter = simplesam.Reader(f)
        for sam in samiter:
            sams[species].append(sam)
```

In [ ]:
```
# Get list of short read names
read_names = [r.id for r in SeqIO.parse(fastqfile, 'fastq')]
# Initialize data frame to count mappings
counts = {species: {k:0 for k in read_names} for species in genomes}
```

```
for species in genomes:
    for name in read_names:
        for i in range(len(sams[species])):
            if sams[species][i].qname == name and sams[species][i].pos > 0:
                # +1 if read is mapped to reference genome
                counts[species][name] += 1

# Convert counts to data frame
df = pd.DataFrame.from_dict(counts)
# Set all counts > 1 to 1 (Avoid counting multiple hits)
df = df.where(df < 1, 1)
# Add Unknown column
df['Unknown'] = (df.sum(axis=1) == 0).astype(int)
# Normalize rows
df = df.div(df.sum(axis=1), axis=0)

df.head()
```

Out[ ]:

	O_uli	S_rotundu	E_coli	Unknown
SRR3656745.86422854	0.0	0.0	0.0	1.0
SRR3656745.64990421	0.0	0.0	0.0	1.0
SRR3656745.91602156	0.0	0.0	0.0	1.0
SRR3656745.79507769	0.0	0.0	0.0	1.0
SRR3656745.38036230	0.0	0.0	0.0	1.0

Results

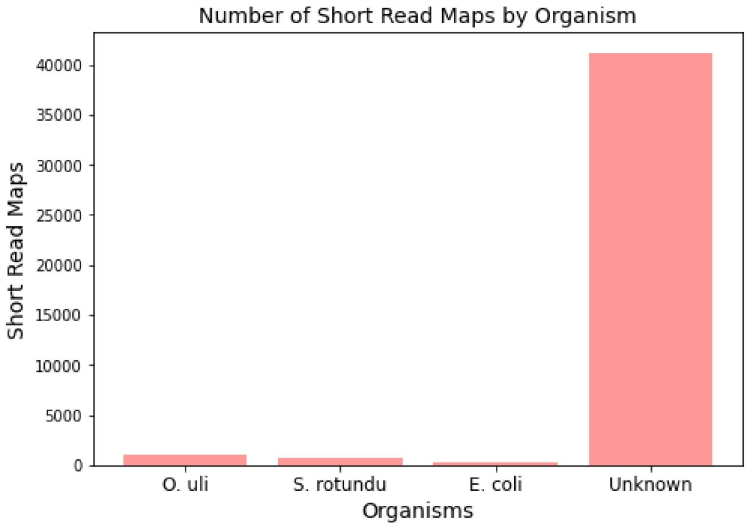
Total Read Counts

Show a bar graph of total read counts assigned to each organism and to the Unknown group. The x axis of the bar graph should show the organisms (including the Unknown group), and the y axis should show the counts of reads mapped to each organism.

In [ ]:
```
total_maps = df.sum().to_list()
species = df.columns.to_list()

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(species, total_maps, color='red', alpha=.4)
ax.set_xticklabels(['O. uli', 'S. rotundu', 'E. coli', 'Unknown'], fontsize=12)
ax.set_xlabel("Organisms", fontsize=14)
ax.set_ylabel("Short Read Maps", fontsize=14)
ax.set_title("Number of Short Read Maps by Organism", fontsize=14)
plt.show()
```

```
/tmp/ipykernel_721/1566673506.py:7: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(['O. uli', 'S. rotundu', 'E. coli', 'Unknown'], fontsize=12)
```



Percent Abundances

Show the percent abundances of the species (**excluding** the Unknown group) as a bar graph. Label the bars with the species names. Make sure you normalize the counts of the reads assigned to each organism by the genome size of that organism. The percentages of the three organisms should add up to 100%.

In [ ]:
```
# Determine genome sizes
sizes = {k: len(r.seq) for k,v in genomes.items() for r in SeqIO.parse(v, 'fasta') }

# Compute normalized percent abundance
size_per_maps = [size/maps for maps, size in zip(total_maps, sizes.values())]
norm_spm = [100*spm/sum(size_per_maps) for spm in size_per_maps]

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(species[:-1], norm_spm, color='red', alpha=.4)
ax.set_xticklabels(['O. uli', 'S. rotundu', 'E. coli'], fontsize=12)
ax.set_xlabel("Organisms", fontsize=14)
ax.set_ylabel("Percent Abundance", fontsize=14)
ax.set_title("Normalized Percent Abbundance by Organism", fontsize=14)
plt.show()
```

```
/tmp/ipykernel_721/2736224981.py:11: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(['O. uli', 'S. rotundu', 'E. coli'], fontsize=12)
```