פרויקט S16 במבוא לגנומיקה וביואינפורמטיקה

מגישה: אביטל אקרמן ת.ז: 208933606

10 אוגוסט, 2022

הקדמה:

בעבודה זו ניסיתי לענות על השאלה :

האם ישנו הבדל במיקרוביום שורשי הצמחים בין מים נקיים למי שפכים שעברו טיהור?

מים נקיים מהווים כ2.8% ממאגרי המים בעולם והם נמצאים ב : אגמים (0.009% ממאגרי המים בעולם), נחלים (0.0001% ממאגרי המים בעולם) , מים אטמוספרים שכוללים אדים , עננים ומשקעים (מהווים כ 0.001% ממאגרי המים בעולם), מי תהום רדודים בקרקע ובאקוויפרים תת קרקעיים (0.31% ממאגרי המים בעולם) וקרחונים בקטבים (מהווים 2.15% ממאגרי המים בעולם). חלק גדול מתוכם לא זמין לצרכים ביולוגיים, בין אם בגלל אחוז מלחים גבוה או הימצאותם בצורה שלא מאפשרת צריכה שלהם (קרחונים או באטמוספרה). לכן המים המתאימים לצריכה הם פחות מ 0.5% מכלל המים בעולם. (1)

מים נקיים משמשים אותנו ואת כלל היצורים החיים לשתייה והכרחיים להישרדותנו. בעולם בו ישנה כמות גדולה של בני אדם וישנה חקלאות אינטנסיבית נוצר מחסור במי שתייה נקיים. לכן מציאת חלופה להם בהיבטים שונים בחיינו כגון בחקלאות יכולה להפחית את הדרישה למים נקיים.

אחת הדרכים לפתור את המחסור במים נקיים היא טיהור מי שפכים. כיום ישנן טכנולוגיות טיהור מים מתקדמות מאוד, ומים כאלו משמשים לצריכה אנושית, לתהליכים תעשייתיים כמקור מי קירור ולחקלאות.

למרות הטכנולוגיות המתקדמות לטיהור מי שפכים והחיוניות של התהליך הזה ישנן בעיות רבות בכך. אחת הבעיות היא הרכב מומסים שונה במים נקיים לעומת מי שפכים מטוהרים (במי קולחין יש ריכוז גבוה יותר של מזהמים אי-אורגניים לדוגמא) (2). שינוי זה יכול להוביל לשינוי ניוטריינטים זמינים לצמחים ולמיקרוביום שלהם. מיקרוביום הצמחים והמגוון שלהם חשובים לניבוי צמיחה והישרדות הצמח. על ידי הפקת מטבוליטים משניים לחיידקים יש השפעה על דרך ההתמודדות של הצמח עם פתוגנים שונים והתמודדות שלו עם תנאי סביבה שונים. המיקרוביום של הצמח מושפע מתנאי הגידול של הצמח. לכן שינוי במי ההשקיה של הצמח יכול להוביל ליבול שונה. (3)

כדי לראות האם אכן ישנו שינוי במיקרוביום של הצמח כתוצאה משינוי במי ההשקיה, טוב להסתכל על המיקרוביום של שורשי הצמח. שורשי הצמחים "רואים" את מי ההשקיה למשך הזמן הארוך ביותר ובאים באינטראקציה מתמדת איתם. לכן, ישנה סבירות גבוהה שאם מי ההשקיה ישפיעו על מיקרוביום הצמח נראה את השינוי הזה במיקרוביום השורשים שלו.

ההשערה שלי הייתה שאכן אראה הבדל במיקרוביום בשתי צורות ההשקיה הללו.

לעבודה זו קיבלתי את חמשת הקבצים הבאים :

- שני קבצי fasta עם כל הריצופים של 16s בכל דוגמא : SS1.fasta_subsanple.fasta, SS2.fasta_subsanple.fasta . כל אחד מהקבצים האלו הופק מריצוף של דגימות שורשי צמחים : אחד שהושקה במים רגילים (נקיים) והשני במי שפכים מטוהרים. כאשר SS1 היא הדוגמא של המים הנקיים וSS2 היא הדוגמא של מי השפכים המטוהרים.
- קובץ שבו יש קוד הפייתון המארגן את גני הs16 לקלאסטרים: parse_cdhit.py
- הדאטא בייס שאשתמש בו לפרויקט (greengenes): 91_otus.fasta.zip
- קובץ טקסט של הטוקסונומיות לפי ה OTU של הדאטא בייס greengene : 91_otus_taxonomy.txt

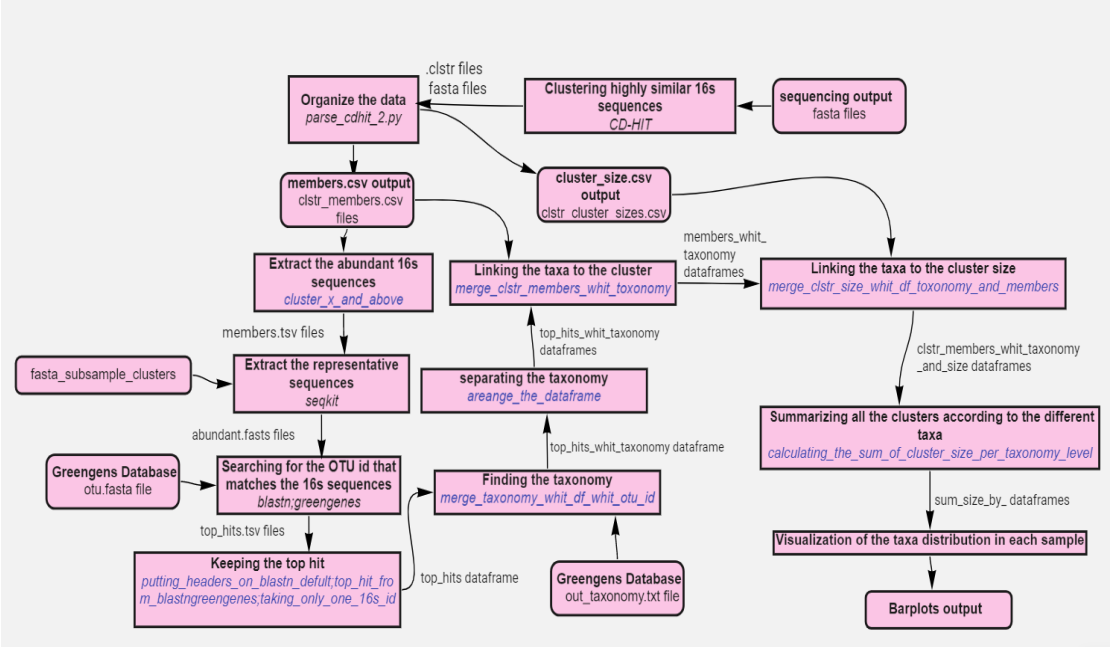
את העבודה ביצעתי בשתי סביבות עבודה :

- ב Ubuntu(סביבת עבודה של לינוקס), את כל פקודות הbash ביצעתי דרכה בכתובת :

/mnt/c/Users/akerm/Downloads/bio_project

- בgoogle colab (סביבת עבודה של פייתון), בסביבה זו ביצעתי את כל האנליזות בפייתון. (מצורף הקובץ הכולל bio_project.ipynb)

The pipeline



פירוט הפעולות שנעשו בפרויקט:

Clustering highly similar 16s sequences

בשני קבצי ה fasta שקיבלתי, היו מספר רב של רצפי 16s. בסביבה ביולוגיות בדרך כלל ישנם מספר מופעים של אותו חיידק עם אותו 16s. כדי שאוכל להסתכל על התמונה הטקסונומית בכל דוגמא, תחילה רציתי להוריד את הכפילויות על ידי קיבוץ של הה16s השונים משני קבצי הfasta שניתנו לי. כדי לעשות זאת השתמשתי בתוכנת cd-hit בגרסא : 4.8.1-2build1 והרצתי אותה בUbuntu. תוכנה זו מקבצת לפי פרמטרים שונים את שורות הטקסט שבfasta כאשר בכל קובץ שקיבלתי, שורה אחת של טקסט שווה לגנום 16s אחד. לכל קלאסטר שנוצר יש גן 16s מייצג.

הפקודות שהרצתי, עם פרמטרי דמיון של 95% על 95% מהאורך של הרצף. הפקודות היו :

- cd-hit -i SS1.fasta_subsample.fasta -o SS1.fasta_subsample_clusters -c 0.95 -n 4 -aL 0.95 -aS 0.95 -sc 1 -T 4 -d 200
- cd-hit -i SS2.fasta_subsample.fasta -o SS2.fasta_subsample_clusters -c 0.95 -n 4 -aL 0.95 -aS 0.95 -sc 1 -T 4 -d 200

שני הקבצים שינתנו לי היו :

- SS1.fasta_subsample.fasta
- SS2.fasta_subsample.fasta

הפלטים של הפקודה הזו היו :

- SS1.fasta_subsample_clusters
- SS1.fasta_subsample_clusters.clstr
- SS2.fasta_subsample_clusters
- SS2.fasta_subsample_clusters.clstr

כאשר הפרמטרים של הפונקציה : (5,4)

- i פרמטר שלאחריו מגיע שם הקובץ שעליו מבצעים את הפקודה (input file).

- o פרמטר שלאחריו מגיע שם הקובץ שלתוכו הפקודה מכניסה את תוצאת ההרצה שלה (output file).
- c זהו פרמטר המראה על סף זהות הרצף, כאשר הזהות של שורת הטקסט הנבדקת שווה או גבוהה מהמספר הרשום אחרי פרמטר זה (במקרה שלנו מתחת ל 95%), השורה נכנסת לקבוצה.
- n מראה על אורך המילה הנבדקת כל פעם (במקרה שלנו זה 4).
- aL כיסוי האליימנט עבור הרצף הארוך יותר (במקרה שלנו 95%).
- aS כיסוי האליימנט עבור הרצף הקצר יותר (במקרה שלנו 95%).
- sc זהו פרמטר שלפיו יש את המיון של הקלסטרים המתקבלים. כאשר הוא 1 המיון הוא לפי גודל הקבוצה , הקבוצה הגדולה ביותר לקטנה ביותר.
- T זהו פרמטר שלפיו נקבע כמות הפעולות/מעבדים שהפקודה יכולה לרוץ עליהם. (במקרה שלנו 4 , בכל ליבה יש 2 threads ולכן יש שימוש ב 2 ליבות לצורך מימוש הפקודה. זאת אומרת שישנה אפשרות להריץ 4 פעולות במקביל).
- d אורך התיאור בקבצי clstr. שיוצאים מהפקודה (המקרה שלנו 200).

כאשר בקבצי ה fasta שנוצרו יש את הרצף 16s המייצג של כל קבוצה, ובקבצי clstr יש את כל הקבוצות אשר מתאימות לפרמטרים שבחרתי ממוינות לפי גודל הקבוצה בסדר יורד.

Organize the data

לאחר קבלת הקבוצות, רציתי לסדר אותן בצורה שמקלה את השימוש במידע, בטבלה. השתמשתי בסקריפט שסופק לי parse_cdhit.py כדי לעשות זאת. שיניתי אותו מעט, על ידי הפיכתו לפונקציה שמאפשרת לקרוא את שני הקבצים ביחד (הקוד החדש בתחתית העמוד ובקובץ המצורף). העתקתי אותו לקובץ חדש בשם parse_cdhit_2.py והרצתי אותו בUbuntu.

```
def make_cluster_table(filename):

    clustnum_pat = re.compile("Cluster\s(\d+)$")
    member_pat = re.compile("\>(.+)\.\.\.")

    with open(filename, "r") as f:
        file = f.readlines()

        outdict = {}
        repr = {}

        for line in file:
            if "Cluster" in line:
                current_clust = re.search(clustnum_pat, line).group(0)
#                print(current_clust)
            else:
                current_member = re.search(member_pat, line).group(1)
#                print(current_member)
                outdict[current_member] = current_clust
            if "*" in line:
                r = line.split(">")[1].split("...")[0]
                repr[current_clust] = r

    outdf = pd.DataFrame.from_dict(outdict, orient = 'index', columns = ['cluster'])
    outdf = outdf.reset_index().rename(columns={"index": "id"})
    reprdf = pd.DataFrame.from_dict(repr, orient = 'index', columns = ['id'])
    reprdf = reprdf.reset_index().rename(columns={"index": "cluster"})
    reprdf['representative'] = True

    mer = outdf.merge(reprdf, on = ['id', 'cluster'], how = 'left')
    mer.fillna(False, inplace = True)

    mer.to_csv(filename + "_members.csv", index = False)

    freq = mer['cluster'].value_counts().to_frame()

    freq.to_csv(filename + "_cluster_sizes.csv", header = None)


make_cluster_table("SS2.fasta_subsample_clusters.clstr")
make_cluster_table("SS1.fasta_subsample_clusters.clstr")
```

<u>שמרתי את הקובץ בעזרת הפקודה :</u>

nano parse_cdhit_2.py

<u>הקוד שהשתמשתי כדי להריץ אותו :</u>

python3 parse_cdhit_2.py

<u>הסקריפט מקבל :</u>

- SS1.fasta_subsample_clusters.clstr
- SS2.fasta_subsample_clusters.clstr

<u>לאחר ההרצה התקבלו הקבצים הבאים :</u>

- SS2.fasta_subsample_clusters.clstr_cluster_sizes.csv
- SS2.fasta_subsample_clusters.clstr_members.csv
- SS1.fasta_subsample_clusters.clstr_cluster_sizes.csv
- SS1.fasta_subsample_clusters.clstr_members.csv

Extract the abundant 16s sequences

כדי שאוכל לראות את תמונת המצב של החיידקים המשפיעים על הצמח, ארצה להסתכל על החיידקים היותר נפוצים. בהנחה שהחיידקים המאוד נדירים לא משפיעים רבות על מיקרוביום הצמח(הנחה שיכולה להיות לא נכונה). בשלב הזה העתקתי את כל קבצי ה members לתוך הgoogle colab בשביל לבצע את האנליזה בצורה נוחה יותר. קראתי אותם בתור דאטאפריימים וכתבתי את הפונקציה cluster_x_and_above .

<u>פונקציה זו מקבלת 3 ארגומנטים :</u>

1. דאטא פריים שהתקבל מקריאת קובץ members:
   - SS2.fasta_subsample_clusters.clstr_members.csv
   - SS1.fasta_subsample_clusters.clstr_members.csv
2. שם הקובץ הסופי שארצה לקבל:
   - df_SS1_members
   - df_SS2_members
3. מספר גנומים בקבוצה שממנו והלאה נרצה לקחת, במקרה שלנו 10. את המספר הזה אפשר לשנות על פי רצונות המחקר, ככל שנוריד אותו נקבל בקובץ הסופי רצפי 16s נדירים יותר. ככל שנעלה מספר זה נקבל רצפי 16s שכיחים יותר, אך עם סכנה גבוהה יותר לאיבוד מידע.

```python
import pandas as pd
import numpy as np


df_SS1_members=pd.read_csv("SS1.fasta_subsample_clusters.clstr_members.csv")
df_SS2_members=pd.read_csv("SS2.fasta_subsample_clusters.clstr_members.csv")

def cluster_x_and_above(df,name,filter_number):
  group=df.groupby(by=['cluster'])
  group_x_and_above=group.filter(lambda x: len(x) >=filter_number)
  df_g = group_x_and_above.query('representative == True')
  df_g=df_g.filter(['id'])
  df_g.to_csv(name,sep='\t', index = False)
  #return df_g



cluster_x_and_above(df_SS1_members ,'df_SS1_members.tsv',10)
cluster_x_and_above(df_SS2_members ,'df_SS2_members.tsv',10)
```

פונקציה זו מקבצת לקבוצות לפי הקלאסטר , מפלטרת לפי מספר הגנומים בקבוצה ולוקחת את הid של רצפי ה representative ורושמת אותם בקובץ tsv ללא האינדקסים.

לאחר ההרצה התקבלו הקבצים הבאים:

- df_SS1_members.tsv
- df_SS2_members.tsv

Extract the representative sequences

בשלב זה הורדתי את הקבצים שהתקבלו בשלב הקודם, לתיקיית העבודה שלי והשתמשתי בפקודת seqkit כדי לשלוף את הרצפים של הrepresentative השכיחים שהוצאתי קודם לכן מקבצי fastaה שקיבלתי בתחילת העבודה.

השתמשתי בפקודות הבאות:

- ./seqkit grep -f df_SS1_members.tsv SS1.fasta_subsample_clusters >SS1_abundant.fasta
- ./seqkit grep -f df_SS2_members.tsv SS2.fasta_subsample_clusters >SS2_abundant.fasta

הקלטים בפקודה הזו:

- df_SS1_members.tsv
- SS1.fasta_subsample_clusters
- df_SS2_members.tsv
- SS2.fasta_subsample_clusters

לאחר ההרצה התקבלו הקבצים הבאים:

- SS1_abundant.fasts
- SS2_abundant.fasts

בכל קובץ כזה יש את רצפים של הrepresentative של הגנומים השכיחים בפורמט fasta, כך שאפשר לעבוד איתם בצורה נוחה יותר.

Searching for the OUT id that matches the 16s sequences

כדי להבין אילו טאקסות יש לנו בדוגמאות, אחפש רצפי 16s דומים במאגר greengenes שקיבלתי בתחילת העבודה. במאגר זה יש רצפי 16s עם הטקסונומיה המתאימה להם. כדי למצוא את הרצפים הדומים לרצפים המייצגים שיש לי בדוגמאות אשתמש ב blastn .

השתמשתי בפקודות הבאות :

- blastn -query SS1_abundant.fasta -subject 91_otus.fasta -num_threads 4 -outfmt 6 -evalue 1e-100 -subject_besthit -max_target_seqs 2 -out SS1_top_hits.tsv
- blastn -query SS2_abundant.fasta -subject 91_otus.fasta -num_threads 4 -outfmt 6 -evalue 1e-100 -subject_besthit -max_target_seqs 2 -out SS2_top_hits.tsv

הקלטים של הפונקציה :

- SS1_abundant.fasta
- SS2_abundant.fasta
- 91_otus.fasta

לאחר ההרצה התקבלו הקבצים הבאים :

- SS1_top_hits.tsv
- SS2_top_hits.tsv

בקבצים אלו יש טבלה של תוצאות הblast כנגד מאגר המידע greengenes של כל אחת מהדוגמאות.

הפרמטרים של הblasn (6,7) :

- query- זהו פרמטר שלאחריו יופיע הקובץ של הגנים שנרצה לעשות להם blast.
- subject- זהו פרמטר שלאחריו יופיע הקובץ של הגנים שאליהם נרצה לעשות את ההשווה, הדאטא בייס שלנו (במקרה greengens).
- num_threads- זהו פרמטר שלפיו נקבע כמות הפעולות/מעבדים שהפקודה יכולה לרוץ עליהם, כפי שהוסבר בהרחבה קודם לכן (במקרה שלנו 4).
- outfmt- פרמטר זה קובע את הדרך/פורמט שבה מוצגת התוצאה, במקרה שלנו פרמטר הזה הוא 6 האומר שהתוצאה של הבלאסט תהיה מוצגת בקובץ tsv.
- evalue- פרמטר זה נותן סף של evalue שרק מערך זה ומטה ילקחו התוצאות. ה evalue אומר את מספר התוצאות הצפויות להתקבל באיכות דומה , שיכולים להימצא במקרה. ככל שערך זה נמוך יותר כך התוצאה שלנו מהימנה יותר, לכן בחרנו בערך נמוך של 1e-100.
- subject_besthit- פרמטר הנותן לשמור את התוצאה הטובה ביותר של הבלאסט.
- max_target_seqs- פרמטר זה אומר כמה תוצאות של ה. alignment לשמור, במקרה שלנו ערך זה הוא 2 מכיוון שלא נרצה עומס במידע (בנוסף הפרמטר הקודם מאפשר לקחת את התוצאות הטובות ביותר ולכן אין צורך ביותר). אך, לקיחת תוצאות מועטות כל כך יכול להוביל להורדת מידע שיכול להיות חשוב.
- out- לאחר פרמטר זה יופיע שם הקובץ שלתוכו יכנסו תוצאות הblastn.

Keeping the top hit

כדי למצוא את התוצאה המתאימה ביותר מבין כל התוצאות שהתקבלו בblast עבור כל רצף 16s העברתי את הקבצים שהתקבלו לgoogle colab. הקבצים שהתקבלו בשלב הקודם הם ללא כותרות לכל עמודה. כדי לסדר זאת (ובכך לתת משמעות לכל עמודה), חיפשתי אילו עמודות יוצאות בפורמט הblastn המתאים (לפי פרמטר outfmt שהגדרתי בפקודת הblastn).

מצאתי שהערכים הדיפלומטיים שהפרמטר הזה מביא הם (8) :

- qseqid- הid של הquery , המספר המזהה שיש ל16s המייצג.
- sseqid- הid של הרצף שאיתו נעשתה ההשוואה (subject) שהוא ה otu id.

- pident- אחוז הדמיון בין ה subject ל query .
- length -אורך ההעמדה (alignment).
- mismatch- כמות חוסר ההתאמות שנוצרו כאשר עשו את ההעמדה.
- gapopen- מספר ה gapים שנוצרות בההעמדה.
- qstart- התחלת ההעמדה על הרצף שלנו(query).
- qend- סוף ההעמדה על הרצף שלנו(query).
- sstart- התחלת ההעמדה על הרצף שאיתו נעשתה ההשוואה (subject).
- send- סוף ההעמדה על הרצף שאיתו נעשתה ההשוואה (subject).
- evalue-ערך הe.
- bitscore- הbitscore של ההעמדה. ערך זה הוא ערך מנורמל לפי ציון ההעמדה, המודד את הדמיון לרצף הquery, ללא תלות באורך הרצף או גודל מאגר המידע.

כדי להכניס סדר בטבלה, בניתי פונקציה putting_headers_on_blastn_defult.

<u>פונקציה זו מקבלת קובץ tsv שהתקבל בשלב הקודם :</u>

- SS1_top_hits.tsv
- SS2_top_hits.tsv

<u>הפלט של הפונקציה הזו, הוא דאטאפריים עם המידע שהיה בקבצים המקוריים בתוספת ל כותרות המתאימות :</u>

- df_SS1_top_hits
- df_SS2_top_hits

```
#Readind the tsv files and puting them into dataframes
df_SS1_top_hits=pd.read_csv("SS1_top_hits.tsv", sep='\t')
df_SS2_top_hits=pd.read_csv("SS2_top_hits.tsv", sep='\t')


def putting_headers_on_blastn_defult(df):
  df.columns=['qseqid' ,'sseqid' ,'pident' ,'length' ,'mismatch' ,'gapopen' ,'qstart' ,'qend' ,'sstart' ,'send' ,'evalue' ,'bitscore']#the default values when doing blastn -outfmt 6
  return df

df_SS1_top_hits=putting_headers_on_blastn_defult(df_SS1_top_hits)
df_SS2_top_hits=putting_headers_on_blastn_defult(df_SS2_top_hits)
```

```
def top_hit_from_blastn(df):
  idx=df.groupby(['qseqid'])['pident'].transform(max)==df['pident']
  df=df[idx]
  return df


df_SS2_top_hits=top_hit_from_blastn(df_SS2_top_hits)
df_SS1_top_hits=top_hit_from_blastn(df_SS1_top_hits)
```

לאחר השימוש בפונקציה הזו רציתי לקחת את התוצאה עם אחוז ההתאמה הגבוה ביותר לכל 16s בדוגמא. תוצאה זו היא בסבירות הגבוהה ביותר ה OTU המתאים לכל חיידק לפי greengenes database. לשם כך , בניתי את הפונקציה top_hit_from_blastn. פונקציה זו מקבלת את הדאטא פריימים שהתקבלו מהפונקציה הקודמת. מקבצת את הדאטאפריים לפי ה query id, ולכל גן כזה לוקחת את אחוז הדמיון הגבוה ביותר.

<u>הקלט של top_hit_from_blastn :</u>

- df_SS1_top_hits
- df_SS2_top_hits

<u>הפלט של top_hit_from_blastn :</u>

- df_SS1_top_hits
- df_SS2_top_hits

לאחר שעשיתי זאת ראיתי שישנן תוצאות לאותו גן 16s שבדוגמא שקיבלו תוצאות עם אחוז דמיון זהה. החלטתי לקחת רק אחת מהתוצאות הללו להמשך. החלטה זו מתבססת על ההנחה שכאשר אחוז הדמיון זהה יש אותו OUT. אך , זה אינו תמיד המצב ופה יכול להיות פספוס מידע.

לשם כך, בניתי פונקציה taking_only_one_16s_id. פונקציה זו מקבלת את הדאטאפריים שהתקבל מהפונקציה הקודמת, עושה איחוד לפי ה query id ולוקחת את האיבר הראשון מכל קבוצה זו.

הקלט של taking_only_one_16s_id:

- df_SS1_top_hits
- df_SS2_top_hits

```
def taking_only_one_16s_id(df):
  group=df.groupby(by=['qseqid'], as_index=False)
  df=group.first()
  return df


df_SS1_top_hits_f=taking_only_one_16s_id(df_SS1_top_hits)
df_SS2_top_hits_f=taking_only_one_16s_id(df_SS2_top_hits)
```

הפלט שלtaking_only_one_16s_id :

- df_SS1_top_hits_f
- df_SS2_top_hits_f

Finding the taxonomy

עכשיו לאחר שיש לי את ה OTU id, אוכל לפי זה למצוא את הטקסונומיה המתאימה לכל גן. בשביל לעשות זאת, הורדתי את קובץ הטקסונומיה שקיבלתי בתחילת העבודה (91_out_taxonomy.txt). קובץ זה מאפשר לקשר בין OTU id לטקסונומיה של כל חיידק. בgoogle colab קראתי את הקובץ הזה כ tsv והכנסתי אותו לדאטאפריים והכנסתי כותרות מתאימות לשם נוחות. לאחר מכן בניתי פונקציה merge_taxonomy_whit_df_whit_otu_id. פונקציה זו מקבלת את הדאטאפריים של הטקסונומיה ואת הדאטאפריים של הOTU id עם אחוז הדמיון הגבוה ביותר שהתקבל מהblast. פונקציה זו מאחדת את שני הדאטאפריימים על פי הOTU id ומורידה עמודות לא רלוונטיות להמשך. פונקציה זו מקבלת דאטאפריים ומתקבל דאטאפריים אחד של כל תוצאות הblast, הOTU id והטקסונומיה המתאימה. בכך מצאתי את הטוקסונומיה המתאימה לכל חיידק בעל שכיחות גבוהה בדוגמא.

קלט הפונקציה:

- 91_out_taxonomy.txt
- df_SS1_top_hits_f
- df_SS2_top_hits_f

```
#Readind the txt as a tsv file and puting them into dataframes whit headers
otu_taxonomy=pd.read_csv("91_otu_taxonomy.txt", sep='\t')
otu_taxonomy.columns=['OTU ID' ,'taxonomy']

def merge_taxonomy_whit_df_whit_otu_id(otu_taxonomy,df_top_hits):
  otu_taxonomy['OTU ID']=otu_taxonomy['OTU ID'].astype(str)
  df_top_hits["sseqid"]=df_top_hits["sseqid"].astype(str)
  df=df_top_hits.merge(otu_taxonomy,left_on="sseqid",right_on="OTU ID",how="left")
  #removing the unnecessary columns
  df=df.drop(["OTU ID","pident","length","mismatch","gapopen","qstart","qend","sstart","send","evalue","bitscore"], axis=1)
  return df

df_SS1_top_hits_whit_taxonomy=merge_taxonomy_whit_df_whit_otu_id(otu_taxonomy,df_SS1_top_hits_f)
df_SS2_top_hits_whit_taxonomy=merge_taxonomy_whit_df_whit_otu_id(otu_taxonomy,df_SS2_top_hits_f)
```

פלט הפונקציה :

- df_SS1_top_hits_whit_taxonomy
- df_SS2_top_hits_whit_taxonomy

Separating the taxonomy

כדי שאפשר יהיה לעבוד בצורה מסודרת יותר עם הטקסונומיות, רציתי להפריד את העמודה המאוחדת של הטקסונומיה לעמודות נפרדות לפי רמות הטקסונומיה השונות. לשם כך בניתי את הפונקציה areange_the_dataframe. פונקציה זו מקבלת דאטאפריים מפרידה את עמודת הטקסונומיה לעמודות נפרדות(אחרי הסימון "_") . בחלק מהטקסונומיות היו חוסרים של טקסות מסוימות. במקרה כזה החלפתי את החוסר בערך nan לשם נוחות.

קלט הפונקציה:

- df_SS1_top_hits_whit_taxonomy
- df_SS2_top_hits_whit_taxonom

```
def areange_the_dataframe(df):
  taxonomy=["kingdom","phylum","class","order","family","genus","species"]
  df[taxonomy] = df['taxonomy'].str.split(';', expand=True)
  for tax in taxonomy:
    df[tax] = df[tax].str.split('_').str[-1]
    #reaplacing the blank str whit nan
    df[taxonomy]=df[taxonomy].replace(r'^\s*$', np.nan, regex=True)
  df=df.drop(["taxonomy"], axis=1)
  return df

df_SS1_top_hits_whit_taxonomy=areange_the_dataframe(df_SS1_top_hits_whit_taxonomy)
df_SS2_top_hits_whit_taxonomy=areange_the_dataframe(df_SS2_top_hits_whit_taxonomy)
```

בסוף התהליך הזה התקבלו 2 דאטאפריימים:

- df_SS1_top_hits_whit_taxonomy
- df_SS2_top_hits_whit_taxonomy

בהם יש את הid של הגן המקורי מהדוגמא, הotu id והטקסונומיה של כל 16s שלעמודות נפרדות לפי דרגות הטקסונומיה השונות.

Linking the taxa to the cluster

כדי להבין בצורה טובה יותר מה מתרחש בדוגמא רציתי לקשר בין הקלסטר של גני ה16s השכיחים שנמצאו בדוגמא עם הטקסונומיה שקיבלתי מהגן המייצג של אותה קבוצה. כדי לעשות כן, הכנסתי את קבצי הmembers שקיבלתי במהלך תחילת העבודה לgoogle colab והמרתי אותם לדאטאפריימים. בניתי פונקציה merge_clstr_members_whit_toxonomy אשר מקבלת דאטאפריים של members ודאטאפריים של גני ה16s המייצגים עם הטקסונומיה שלהם. פונקציה זו מאחדת בין שני דאטאפריימים אלו על סמך id של ה16s שהתקבל בדוגמא. בסוף, מתקבל דאטאפריים מאוחד שמכיל את מספר ה cluster של כל גן מייצג עם הטקסונומיה המתאימה והid.

```
ss1_clstr_members=pd.read_csv("SS1.fasta_subsample_clusters.clstr_members.csv")
ss2_clstr_members=pd.read_csv("SS2.fasta_subsample_clusters.clstr_members.csv")


def merge_clstr_members_whit_toxonomy(df_top_hits_whit_taxonomy,df_clstr_members):
  df=df_clstr_members.merge(df_top_hits_whit_taxonomy,left_on="id",right_on="qseqid",how="right")
  df_only_relevent_col=df[["qseqid","cluster","sseqid","kingdom","phylum","class","order","family","genus","species"]]
  df_merged=pd.merge(df_clstr_members, df_only_relevent_col,on=["cluster"],how="right")
  return df_merged


df_SS1_clstr_members_whit_taxonomy=merge_clstr_members_whit_toxonomy(df_SS1_top_hits_whit_taxonomy,ss1_clstr_members)
df_SS2_clstr_members_whit_taxonomy=merge_clstr_members_whit_toxonomy(df_SS2_top_hits_whit_taxonomy,ss2_clstr_members)
```

<u>הקלט של הפונקציה :</u>

- df_SS1_top_hits_whit_taxonomy
- ss1_clstr_members
- df_SS2_top_hits_whit_taxonomy
- ss2_clstr_members

<u>הפלט של הפונקציה :</u>

- df_SS1_clstr_members_whit_taxonomy
- df_SS2_clstr_members_whit_taxonomy

אציין שאת הטקסונומיה מצאתי לגנים בעלי השכיחות הגבוהה (10 members ומעלה בcluster שהתקבל). קובץ members מכיל את כל הגנומים, גם של קבוצות עם פחות מ 10 members בתוכה. לכן גודל הדאטאפריים לאחר האיחוד קטן יותר מאשר קובץ זה ויכיל רק את הטקסונומיה של אותם גנומים.

Linking the taxa to the cluster size

כדי שאוכל לכמת את נוכחות כל טקסה בדוגמאות שלי ארצה לאחד את גודל הcluster שנמצא ב קבצי "…_cluster_sizes.csv" עם הדאטאפריימים שקיבלתי בחלק הקודם. תחילה קראתי את שני קבצי ה"…_cluster_sizes.csv" כדאטאפריימים (ss2_clstr_sizes, ss1_clstr_sizes) והוספתי כותרות מתאימות לעמודות לשם נוחות.

לאחר מכן בניתי את הפונקציה merge_clstr_size_whit_df_toxonomy_and_members.
<u>פונקציה זו מקבלת שני ארגומנטים :</u>

1. df_clstr_members_whit_taxonomy שקיבלתי בשלב הקודם :
   - df_SS1_clstr_members_whit_taxonomy
   - df_SS2_clstr_members_whit_taxonomy
2. שני הדאטאפריימים שיצרתי מקבצי ה "…_cluster_sizes.csv" :
   - ss1_clstr_sizes
   - ss2_clstr_sizes

פונקציה זו מאחדת את שני הדאטאפריימים האלו דרך עמודת ה cluster. היא יוצרת דאטאפריים חדש המכיל גם את גודל כל קלסטר. בנוסף לכך, היא מסדרת את הדאטאפריים שיהיה נוח יותר להסתכל עליו.

<u>הפלט של הפונקציה הזו :</u>

- df_SS1_clstr_members_whit_taxonomy_and_size
- df_SS2_clstr_members_whit_taxonomy_and_size

```
#Read the files so that the first line is not the column headers. Then insertion of appropriate headings
ss1_clstr_sizes=pd.read_csv("SS1.fasta_subsample_clusters.clstr_cluster_sizes.csv", header=None)
ss1_clstr_sizes.columns=['cluster' ,'size']
ss2_clstr_sizes=pd.read_csv("SS2.fasta_subsample_clusters.clstr_cluster_sizes.csv", header=None)
ss2_clstr_sizes.columns=['cluster' ,'size']

def merge_clstr_size_whit_df_toxonomy_and_members(df_clstr_members_whit_taxonomy,df_clstr_sizes):
  df=pd.merge(df_clstr_members_whit_taxonomy, df_clstr_sizes,on=["cluster"],how="left")
  #Beautify the created data frame
  second_column = df.pop('sseqid')
  forth_column = df.pop('size')
  df.insert(1, 'sseqid', second_column )
  df.insert(3, 'size', forth_column )
  df=df.sort_values("cluster")
  df=df.reset_index(drop=True)
  return df
df_SS1_clstr_members_whit_taxonomy_and_size=merge_clstr_size_whit_df_toxonomy_and_members(df_SS1_clstr_members_whit_taxonomy,ss1_clstr_sizes)
df_SS2_clstr_members_whit_taxonomy_and_size=merge_clstr_size_whit_df_toxonomy_and_members(df_SS2_clstr_members_whit_taxonomy,ss2_clstr_sizes)
```

Summarizing all the clusters according to the different taxa

לאחר מכן, רציתי לראות את הנוכחות של כל טקסה בדוגמא. כדי לענות על השאלה אילו טקסות נמצאות בכל דוגמא ובאיזה כמות. בכך ניתן להסיק על השפעת מי ההשקיה על מגוון המיקרוביום והכמות שלו. כדי לעשות זאת בניתי פונקציה
. calculating_the_sum_of_cluster_size_per_taxonomy_level

<u>פונקציה זו מקבלת שני ארגומנטים :</u>

1. את הדאטאפריים שקיבלתי בשלב הקודם :
   - df_SS1_clstr_members_whit_taxonomy_and_size
   - df_SS2_clstr_members_whit_taxonomy_and_size
2. את שם הטאקסה שנרצה לקבץ ולסכום לפיה :
   - Phylum
   - Class

פונקציה זו מקבצת את הדאטאפריים לפי רמת הטקסונומיה הרצויה וסוכמת את גודל הקלסטרים של אותה טקסה.

<u>הפלט של הפונקציה הזו :</u>

- df_SS1_sum_size_by_phylum
- df_SS2_sum_size_by_phylum
- df_SS1_sum_size_by_class
- df_SS2_sum_size_by_class

```python
def calculating_the_sum_of_cluster_size_per_taxonomy_level(df,taxonomy_level):
  group=df.groupby([taxonomy_level], as_index=False)
  df= group["size"].sum()
  return df

df_SS1_sum_size_by_phylum=calculating_the_sum_of_cluster_size_per_taxonomy_level(df_SS1_clstr_members_whit_taxonomy_and_size,"phylum")
print("The sum of the clusters size according to phylum in SS1 is :")
print(df_SS1_sum_size_by_phylum)
print("-----------------------------------------------------------------------------")
df_SS1_sum_size_by_class=calculating_the_sum_of_cluster_size_per_taxonomy_level(df_SS1_clstr_members_whit_taxonomy_and_size,"class")
print("The sum of the clusters size according to class in SS1 is :")
print(df_SS1_sum_size_by_class)
print("-----------------------------------------------------------------------------")

df_SS2_sum_size_by_phylum=calculating_the_sum_of_cluster_size_per_taxonomy_level(df_SS2_clstr_members_whit_taxonomy_and_size,"phylum")
print("The sum of the clusters size according to phylum in SS2 is :")
print(df_SS2_sum_size_by_phylum)
print("-----------------------------------------------------------------------------")
df_SS2_sum_size_by_class=calculating_the_sum_of_cluster_size_per_taxonomy_level(df_SS2_clstr_members_whit_taxonomy_and_size,"class")
print("The sum of the clusters size according to class in SS2 is :")
print(df_SS2_sum_size_by_class)
print("-----------------------------------------------------------------------------")
```

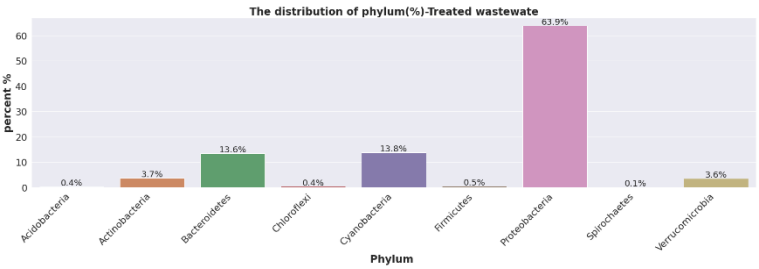
Visualization of the taxa distribution in each sample

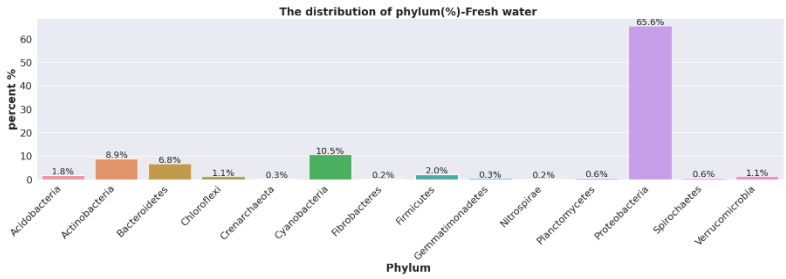
לאחר שקיבלתי את סכום כל הטקסות השונות לפי הרמה הטקסונומית שרציתי, נשאר רק לעשות ויזואליזציה של התפלגות כל טקסה בכל דוגמא. השתמשתי ב matplotlib and seaborn כדי לעשות זאת.

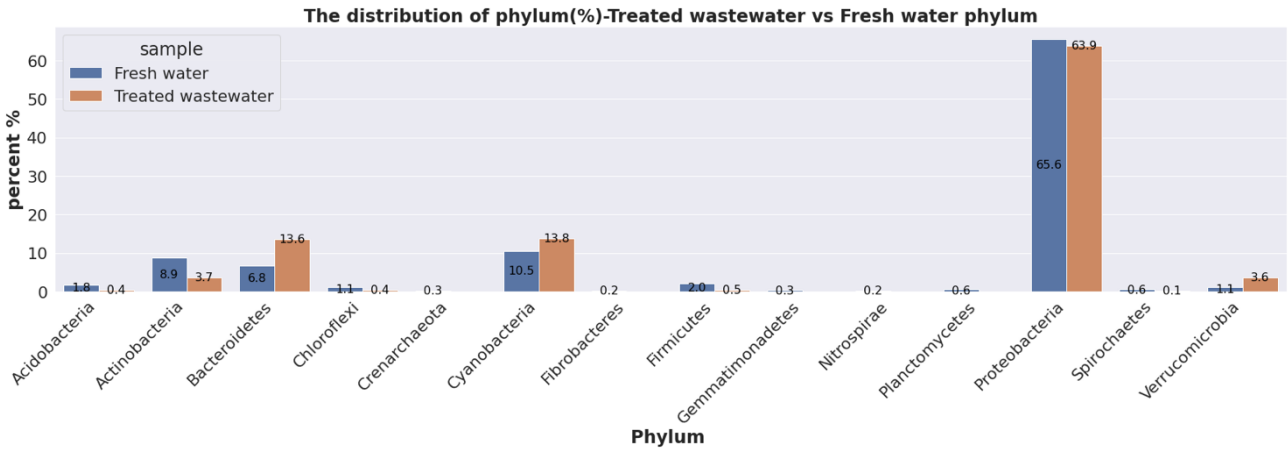
כדי לעשות את ההשוואה יותר אינטואיטיבית ביצעתי את הגרפים גם בסכום וגם באחוזים מכלל הדוגמא.

ברמת הphylum :

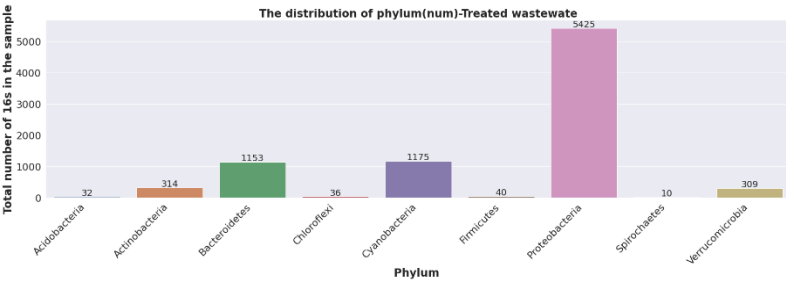
מבחינת אחוזים :

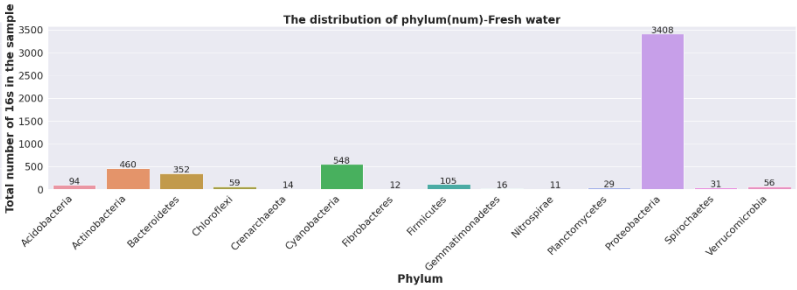


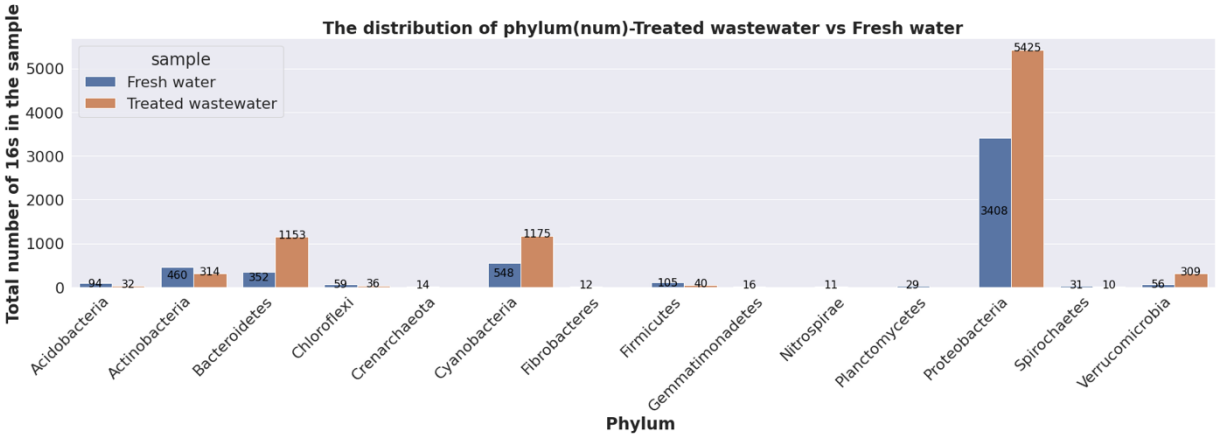




מבחינה מספרית :





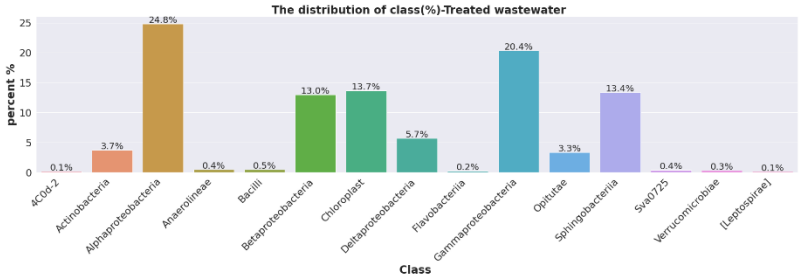


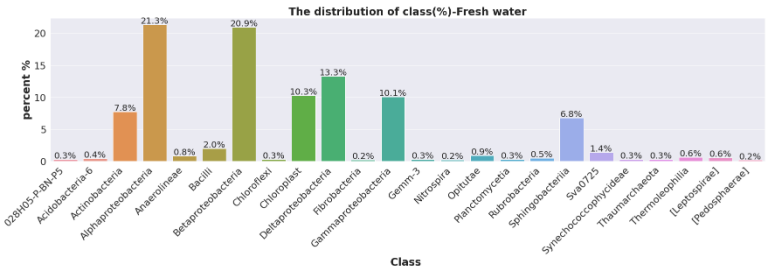
ניתן לראות שבהשקיה במי שפכים מטוהרים יש מגוון קטן יותר של המיקרוביום ברמת ה phylum מאשר כאשר ההשקיה הייתה במים נקיים. דבר זה מתיישב עם העובדה שמי שפכים מטוהרים, לא מנוקים עד הסוף ממזהמים ולכן מגוון קטן יותר של מיקרואורגניזמים יכולים לחיות בסביבה כזו.

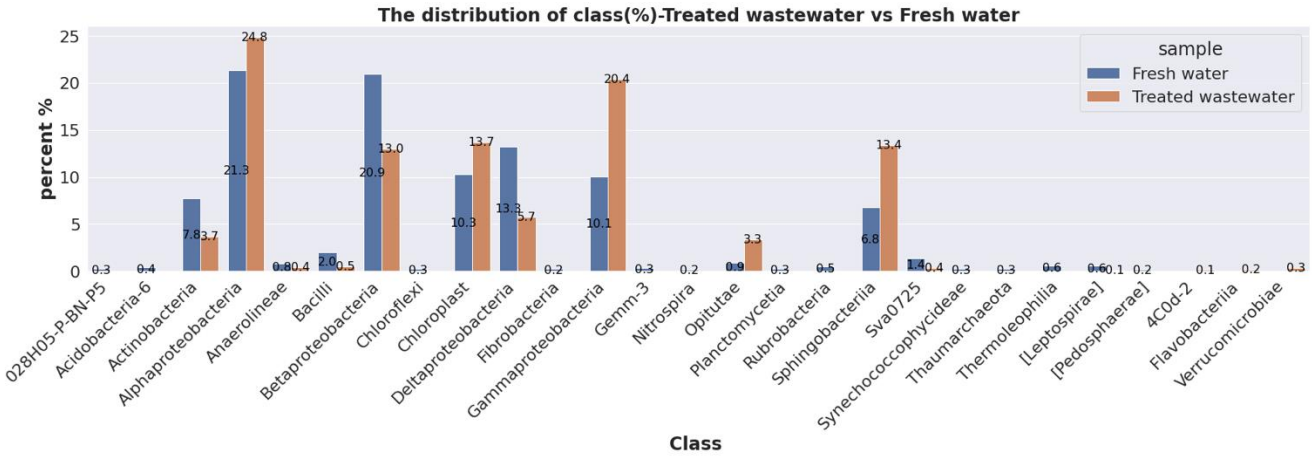
בנוסף, מספר גנומי 16s שהיו בדוגמת מי השפכים המטוהרים הייתה 8,494 בעוד בדוגמת המים הנקיים 5,195. ניתן לראות שהphylum הדומיננטיים בהשקיה של מים נקיים נמצאו גם בדוגמת מי השפכים המטוהרים. ברוב הphylum שנמצאים בדוגמת מי השפכים המטוהרים, מספר הגנומים הנמצאים בכל phylum כזה הוא גבוה יותר מאשר בדוגמת המים הנקיים.אפשר להניח שאלו הן קבוצות שמותאמות יותר לסביבה עם מזהמים. עצם גדילת מספר המיקרואורגניזמים בדוגמת מי השפכים בכ63% מרמז על כך שקבוצות המיקרואורגניזמים שהיו באחוז קטן בדוגמת המים הנקיים, מנעה מהקבוצות הדומיננטיות לגדול ולהשתלט על המיקרוביום של שורשי הצמח.

ברמת הclass :

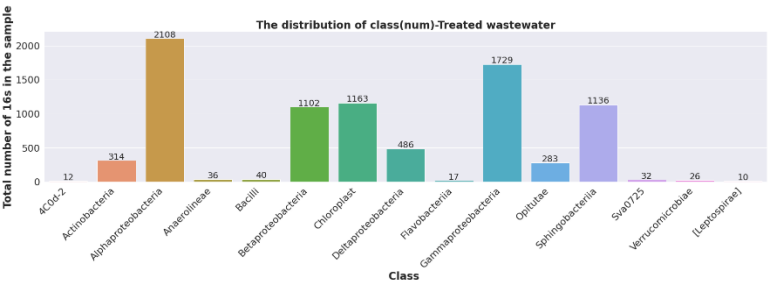
מבחינה אחוזית :

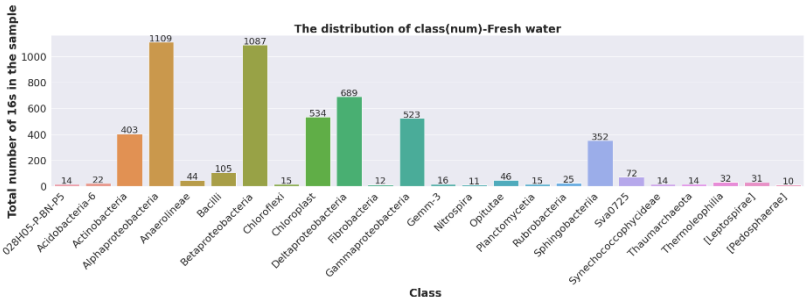


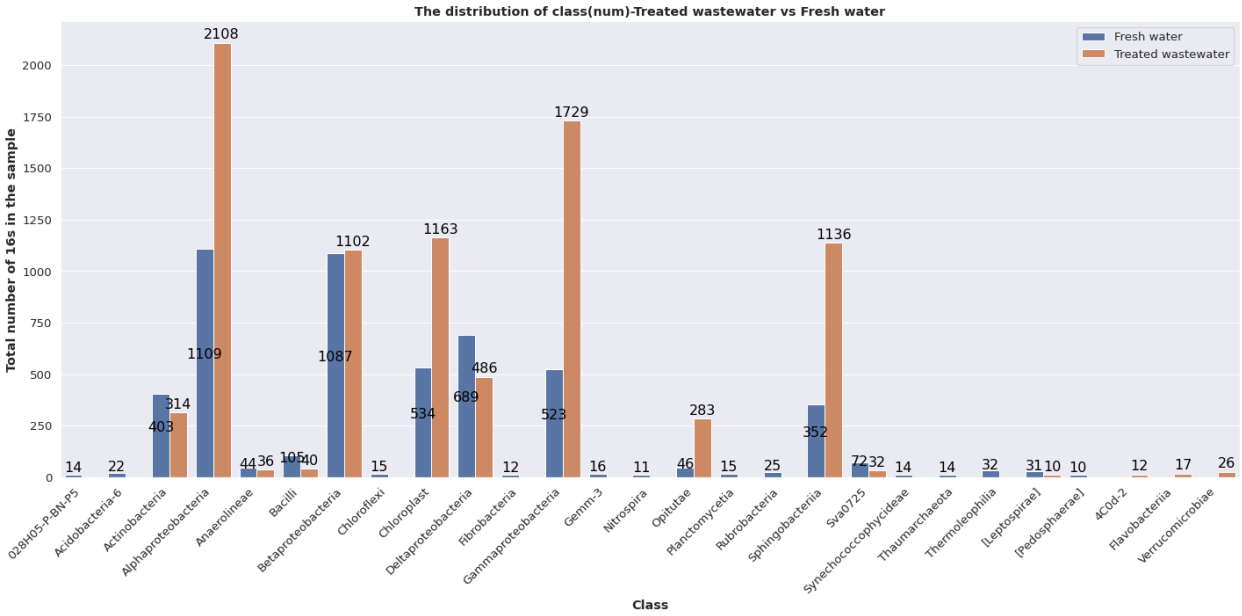




מבחינה מספרית :







ניתן לראות מגמה דומה למה שראינו ברמה הטקסונומית של הphylum. תוצאות אלו יכולות לרמוז על יחסי הגומלין שמתרחשים בין שורשי הצמח לחיידקים בטקסה, ובין החיידקים הללו לחיידקים מטקסות אחרות.

דיון ומסקנות

במהלך העבודה הנחתי כמה הנחות:

1. חיידקים עם אחוז דמיון של 95% ומעלה בגן ה 16s שלהם הם אותו החיידק- זוהי הנחה שאינה בהכרח נכונה. חיידקים יכולים להיות בעלי דמיון גבוה מכך בגן זה ולהיות שונים. קיבוץ של כל החיידקים ולקיחה של גן מייצג אחד יכולה לגרום לאיבוד מידע חשוב.
2. גני 16s השכיחים הם החשובים להבנת המיקרוביום של שורשי הצמחים- הדרך שבה ביצעתי את המחקר התייחסה לגני ה16s השכיחים (10 ומעלה מופעים), דבר הנותן תמונה יחסית טובה של המתרחש בשורשי הצמח. אך זו אינה תמונה מדויקת, גם למספר מועט של מיקרואורגניזמים יכולה להיות השפעה כלשהי על המערכת האקולוגית. לכן, בהנחה זו יש איבוד מידע
3. מאגר greengenes שסופק לי הוא עדכני ומקיף- השתמשתי במאגר הסופק לי לצורך הוצאת הטקסות. אך, ראיתי שבחלק מהטקסות חסרות דרגות טקסונומיות מסוימות. דבר זה יכול להתרחש עקב מאגר מידע לא עדכני/מקיף. מאגרי המידע מתעדכנים תמידית, עצומים ויכול להיות שהמאגר שסופק לי הוא לא המעודכן ביותר. בנוסף בכך שחסרות דרגות טוקסונומיות "נמוכות" יותר אין אפשרות לזהות במדויק כל חיידק. מה שלא מאפשר להבין בצורה עמוקה את השינוי במגוון הטקסונומי.
4. המידע שסופק לי מדויק ונקי מספיק- קיבלתי את קבצי הfasta עם ריצוף של גנומי ה16s מוכנים. לא ראיתי את תהליך הדימולטיפלקסינג ואת איכות הקריאות שהתקבלו. יכול להיות שהקריאות שהתקבלו לא איכותיות מספיק ונדרש ניקוי שלהן, או לקיחת דוגמאות אחרות לצורך זיהוי נכון ומדויק יותר של הטאקסות השונות.

שינוי של כל אחת מהנחות האלו משנה את טיב העבודה לניתוח הנתונים ויכול לתת שינוי משמעותי לתוצאות המתקבלות.

מכיוון שקיבלתי רק 2 דוגמאות (דוגמא אחת מכל טיפול) לא אוכל להסיק מסקנה גורפת על השפעת מי ההשקיה על מיקרוביום שורשי הצמחים. נדרשות חזרות רבות וצמחים ממינים שונים כדי לנסות לקבוע משהו בצורה מדויקת יותר.

אך מהנתונים שקיבלתי אפשר היה לראות שאכן היה שינוי במגוון ובכמות המיקרוביום בין שני הטיפולים. ניתן לראות שהמגוון הטקסונומי בהשקיה במים נקיים הוא גדול יותר, ישנם טקסות נדירות יותר שלא נמצאות בדוגמת מי הקולחין. אפשר להסביר זאת על ידי כך שמי השפכים המטוהרים עדיין מכילים שאריות של מזהמים אי-אורגניים שיכולים לפגוע במיקרואורגניזמים, במיוחד במיקרואורגניזמים הנדירים "שבקושי מצליחים לשרוד" גם כך בסביבת שורשי הצמח.

אך בשונה מכך, כמות החיידקים בדוגמת מי ההשקיה הנקיים קטנה יותר מאשר בטיפול במי שפכים שעברו טיהור. דבר זה יכול להיות מוסבר בכמה דרכים:

- המיקרואורגניזמים הנדירים "ריסנו" את הגדילה של המיקרואורגניזמים השכיחים. בכך שהם "ירדו" המיקרואורגניזמים השכיחים יכלו להתרבות בצורה טובה יותר.
- המזהמים האי-אורגניים שנמצאים במי הקולחין, עוזרים בגדילת חלק מהמיקרואורגניזמים השכיחים ולא מפריעים לשאר יותר מידי.

כדי לדעת את ההשפעה המדויקת של מי הקולחין , יש לבדוק את הרכבם מבחינה כימית , לעשות ניסוי גדול יותר בו מתבצע איסוף לא רק של גנומי ה16s אלא גם של הmRNA (הגנים הפעילים) בכל דוגמא.

ביבליוגרפיה

(1)https://science.jrank.org/pages/2857/Freshwater.html

(2) https://magazine.isees.org.il/?p=16361

(3) Anal, A. K. D., Rai, S., Singh, M., & Solanki, M. K. (2020). Plant mycobiome: Current research and applications. Phytobiomes: Current insights and future vistas, 81-104.

(4) http://www.bioinformatics.org/cd-hit/cd-hit-user-guide.pdf

(5) https://vcru.wisc.edu/simonlab/bioinformatics/programs/cd-hit/cdhit-user-guide.pdf

(6) https://www.ncbi.nlm.nih.gov/books/NBK279684/

(7) https://open.oregonstate.education/computationalbiology/chapter/command-line-blast/

(8) https://www.metagenomics.wiki/tools/blast/blastn-output-format-6