

Additional_Efforts

March 7, 2024

0.1 Additional Reproducibility Efforts

Importing necessary library for data analysis for .csv and .tsv files; **pandas** & **numpy** All the supplementary tables (STable*) downloaded from the Zenodo repository [1] of the original article [2].

[1] A. Gavriilidou, "Compendium of specialized metabolite biosynthetic diversity encoded in bacterial genomes". Zenodo, Apr. 15, 2022.(<https://doi.org/10.5281/zenodo.5159210>).

[2] Gavriilidou, A., Kautsar, S.A., Zaburannyi, N. et al. Compendium of specialized metabolite biosynthetic diversity encoded in bacterial genomes. Nat Microbiol 7, 726–735 (2022). (<https://doi.org/10.1038/s41564-022-01110-2>)

```
[1]: import numpy as np
import pandas as pd
```

```
[2]: STable1: pd.DataFrame = pd.read_csv("STable1_all_genomes_info.tsv", sep="|")
STable2: pd.DataFrame = pd.read_csv("STable2_BiG-SLICE_t0.4_GCF_assignment.
↳csv", sep=",")
STable3: pd.DataFrame = pd.read_csv("STable3_BiG-SLICE_t0.5_GCF_assignment.
↳csv", sep=",")
STable4: pd.DataFrame = pd.read_csv("STable4_BiG-SLICE_t0.6_GCF_assignment.
↳csv", sep=",")
STable5: pd.DataFrame = pd.read_csv("STable5_BiG-SLICE_t0.7_GCF_assignment.
↳csv", sep=",")
```

```
[3]: # Rename the column "bgc_ids" to "bgc_id"
STable1 = STable1.rename(columns={"bgc_ids": "bgc_id"})

# Split multiple IDs in the "bgc_id" column into separate rows
STable1["bgc_id"] = STable1.bgc_id.str.split(",")
STable1 = STable1.explode("bgc_id").reset_index(drop=True)

# Convert the "bgc_id" column to integer values
STable1["bgc_id"] = STable1["bgc_id"].astype("int64")
```

For every threshold **pd.merge** applied. Taxonomy information is selected from taxonomy column of each dataframe. The streptomyces genus level is displayed as an output, the following question is asked;

- From the figure 1_A the streptomyces value counts are known (see Figure_1.ipynb), if the following filtering algorithm utilized do we get the same results?

Figure 1_A data;

	<i>Genus</i>	<i>Number of GCFs</i>	<i>Threshold (T)</i>
0	<i>Streptomyces</i>	7294	$t=0.4$
9	<i>Streptomyces</i>	5720	$t=0.5$
18	<i>Streptomyces</i>	4360	$t=0.6$
27	<i>Streptomyces</i>	2889	$t=0.7$

```
[4]: def filter_and_count_streptomyces(
    stable1: pd.DataFrame,
    stable2: pd.DataFrame,
    threshold_table_name: str = "",
    save_to_file: bool = True,
) -> tuple[pd.DataFrame, pd.DataFrame, pd.Series]:
    """
    Filters the combined DataFrame based on Streptomyces genus,
    counts GCF IDs, and returns the merged, filtered, and taxonomy DataFrames.

    Args:
        stable1 (pd.DataFrame): DataFrame containing genomic information.
        stable2 (pd.DataFrame): DataFrame containing GCF information.
        threshold_table_name (str, optional): Name prefix for the output file.
        ↪ Defaults to "".
        save_to_file (bool, optional): Whether to save the results to a CSV
        ↪ file. Defaults to True.

    Returns:
        tuple[pd.DataFrame, pd.DataFrame, pd.Series]: A tuple containing:
        - pd.DataFrame: The merged DataFrame.
        - pd.DataFrame: The filtered DataFrame containing Streptomyces
        ↪ entries.
        - pd.Series: Counts of GCF IDs for Streptomyces entries.
    """

    # Merge the two DataFrames based on the 'bgc_id' column
    merged_df = stable1.merge(stable2, on='bgc_id')

    # Split the 'taxonomy' column, fill missing values, and set the column name
    taxonomy = merged_df['taxonomy'].str.split(",", expand=True).fillna("")
    taxonomy.name = f"taxonomy_{threshold_table_name}"

    # Filter the DataFrame for entries where the sixth column ('taxonomy[5]')
    ↪ is "Streptomyces"
```

```

streptomyces_df = merged_df[taxonomy[5] == "Streptomyces"]

# Count occurrences of GCF IDs in the filtered DataFrame
streptomyces_gcf_count = streptomyces_df['gcf_id'].value_counts()

# Save the Series to a CSV file (optional)
if save_to_file:
    filename = f"{threshold_table_name}_streptomyces_gcf_count.csv"
    streptomyces_gcf_count.to_csv(filename)

# Return all relevant DataFrames and Series
return merged_df, taxonomy, streptomyces_gcf_count

```

```

[5]: # Assuming you have STable1 and STable2 DataFrames
thresholds_4_gcfs, taxonomy_4, streptomyces_4_count = □
    ↪ filter_and_count_streptomyces(STable1, STable2, "t0.4")

# Repeat the process for other thresholds (STable3, STable4, STable5)
thresholds_5_gcfs, taxonomy_5, streptomyces_5_count = □
    ↪ filter_and_count_streptomyces(STable1, STable3, "t0.5")

thresholds_6_gcfs, taxonomy_6, streptomyces_6_count = □
    ↪ filter_and_count_streptomyces(STable1, STable4, "t0.6")

thresholds_7_gcfs, taxonomy_7, streptomyces_7_count = □
    ↪ filter_and_count_streptomyces(STable1, STable5, "t0.7")

```

```

[6]: len(streptomyces_4_count), len(streptomyces_5_count), len(streptomyces_6_count), len(streptomyces_7_count)

```

```

[6]: (8703, 6798, 5136, 3363)

```

When inspected;

Figure 1_A data from the applied algorithm;

Original Data

Reproduced Data

	<i>Genus</i>	<i>Number of GCFs</i>	<i>Threshold (T)</i>
0	<i>Streptomyces</i>	7294	$t=0.4$
9	<i>Streptomyces</i>	5720	$t=0.5$
18	<i>Streptomyces</i>	4360	$t=0.6$
27	<i>Streptomyces</i>	2889	$t=0.7$

	<i>Genus</i>	<i>Number of GCFs</i>	<i>Threshold (T)</i>
0	<i>Streptomyces</i>	8703	$t=0.4$

	<i>Genus</i>	<i>Number of GCFs</i>	<i>Threshold (T)</i>
9	<i>Streptomyces</i>	6798	$t=0.5$
18	<i>Streptomyces</i>	5136	$t=0.6$
27	<i>Streptomyces</i>	3363	$t=0.7$

[7]: `len(streptomyces_4_count)-7294,len(streptomyces_5_count)-5720,len(streptomyces_6_count)-4360,len(streptomyces_7_count)-3363`

[7]: (1409, 1078, 776, 474)

[8]: `len(streptomyces_4_count)-len(streptomyces_5_count),len(streptomyces_5_count)-len(streptomyces_6_count),len(streptomyces_6_count)-len(streptomyces_7_count)`

[8]: (1905, 1662, 1773)

[9]: `1409-1078, 1078-776, 776-474`

[9]: (331, 302, 302)

[10]: `len(STable2)-len(thresholds_4_gcfs),len(STable3)-len(thresholds_5_gcfs),len(STable4)-len(thresholds_6_gcfs),len(STable5)-len(thresholds_7_gcfs)`

[10]: (764, 37, 37, 37)

When inspected;

Figure 1_A data from the applied algorithm;

Original Data vs Reproduced Data

Extra Operations

	<i>Genus</i>	<i>Number of GCFs Differ</i>	<i>Threshold (T)</i>
$diff_1$	<i>Streptomyces</i>	1409	$t=0.4$
$diff_2$	<i>Streptomyces</i>	1078	$t=0.5$
$diff_3$	<i>Streptomyces</i>	776	$t=0.6$
$diff_4$	<i>Streptomyces</i>	474	$t=0.7$

<i>Operation</i>	<i>Values of the output</i>
$t_{0.4} - t_{0.5}, t_{0.5} - t_{0.6}, t_{0.6} - t_{0.7}$	(1905, 1662, 1773)
\$ diff_1 - diff_2 , diff_2 - diff_3 , diff_3 - diff_4 \$	(331, 302, 302)
Data Lost From INNER MERGE OPERATION $\{t=0.4,0.5,0.6,0.7\}$	(764, 37, 37, 37)

1 Taxonomy resolution

```
[11]: def taxonomy_resolution(taxonomy_file: pd.DataFrame) -> pd.DataFrame:
      """
      Calculates the number of unique taxa at each taxonomic level (phylum,
      ↪class, order, family, genus, species)
      from a given taxonomy file, and returns the results as a DataFrame.

      Args:
          taxonomy_file (pd.DataFrame): DataFrame containing taxonomic
          ↪information,
          have columns representing different taxonomic levels (e.g., phylum,
          ↪class, order, etc.).

      Returns:
          pd.DataFrame: A new DataFrame with the following columns:
          - "phylum_taxa_name": Names of unique phyla
          - "phylum_taxa_count": Number of occurrences for each phylum
          - "class_taxa_name": Names of unique classes
          - "class_taxa_count": Number of occurrences for each class
          - "order_taxa_name": Names of unique orders
          - "order_taxa_count": Number of occurrences for each order
          - "family_taxa_name": Names of unique families
          - "family_taxa_count": Number of occurrences for each family
          - "genus_taxa_name": Names of unique genera
          - "genus_taxa_count": Number of occurrences for each genus
          - "species_taxa_name": Names of unique species
          - "species_taxa_count": Number of occurrences for each species

      """

      # Count unique taxa at each level (phylum, class, order, family, genus,
      ↪species)
      phylum_taxa = taxonomy_file[1].value_counts()
      class_taxa = taxonomy_file[2].value_counts()
      order_taxa = taxonomy_file[3].value_counts()
      family_taxa = taxonomy_file[4].value_counts()
      genus_taxa = taxonomy_file[5].value_counts()
      species_taxa = taxonomy_file[6].value_counts()

      # Determine the minimum list length to ensure equal representation across
      ↪all levels
      max_list_number = min(len(phylum_taxa), len(class_taxa), len(order_taxa),
      ↪len(family_taxa), len(genus_taxa), len(species_taxa))

      # Truncate each Series to the minimum length to maintain consistency
      phylum_taxa = phylum_taxa[:max_list_number]
      class_taxa = class_taxa[:max_list_number]
```

```

order_taxa = order_taxa[:max_list_number]
family_taxa = family_taxa[:max_list_number]
genus_taxa = genus_taxa[:max_list_number]
species_taxa = species_taxa[:max_list_number]

# Create a DataFrame from the Series data
taxonomy_resolution = pd.DataFrame({
    "phylum_taxa_name": list(phylum_taxa.index),
    "phylum_taxa_count": list(phylum_taxa.values),
    "class_taxa_name": list(class_taxa.index),
    "class_taxa_count": list(class_taxa.values),
    "order_taxa_name": list(order_taxa.index),
    "order_taxa_count": list(order_taxa.values),
    "family_taxa_name": list(family_taxa.index),
    "family_taxa_count": list(family_taxa.values),
    "genus_taxa_name": list(genus_taxa.index),
    "genus_taxa_count": list(genus_taxa.values),
    "species_taxa_name": list(species_taxa.index),
    "species_taxa_count": list(species_taxa.values),
})

# Save the DataFrame to a CSV file with a descriptive filename
taxonomy_resolution.to_csv(f"taxonomy_resolution_{taxonomy_file.name}.csv")

# Return the DataFrame for further use
return taxonomy_resolution

```

```

[12]: taxonomy_resolution_4: pd.DataFrame =
↳ taxonomy_resolution(taxonomy_file=taxonomy_4)

taxonomy_resolution_5: pd.DataFrame =
↳ taxonomy_resolution(taxonomy_file=taxonomy_5)

taxonomy_resolution_6: pd.DataFrame =
↳ taxonomy_resolution(taxonomy_file=taxonomy_6)

taxonomy_resolution_7: pd.DataFrame =
↳ taxonomy_resolution(taxonomy_file=taxonomy_7)

```

```

[13]: taxonomy_resolution_4.head()

```

```

[13]:   phylum_taxa_name  phylum_taxa_count  class_taxa_name  class_taxa_count  \
0   Proteobacteria          565024  Gammaproteobacteria          507067
1  Actinobacteriota          264466    Actinobacteria          262184
2    Firmicutes            229209        Bacilli          228404
3   Bacteroidota           32879  Alphaproteobacteria          57906
4   Firmicutes_A           28999    Bacteroidia          30909

```

	order_taxa_name	order_taxa_count	family_taxa_name	family_taxa_count	\
0	Enterobacterales	202288	Enterobacteriaceae	170657	
1	Mycobacteriales	180105	Mycobacteriaceae	164511	
2	Pseudomonadales	170045	Pseudomonadaceae	124741	
3	Burkholderiales	100727	Burkholderiaceae	90287	
4	Lactobacillales	94986	Streptococcaceae	75486	

	genus_taxa_name	genus_taxa_count	species_taxa_name	\
0	Mycobacterium	105197		
1	Pseudomonas	75072	Mycobacterium tuberculosis	
2	Staphylococcus	74118	Pseudomonas aeruginosa	
3	Streptococcus	74113	Staphylococcus aureus	
4	Escherichia	68537	Streptococcus pneumoniae	

	species_taxa_count
0	115269
1	93107
2	74268
3	63582
4	50516

```
[14]: taxonomy_resolution_5.head()
```

```
[14]:
```

	phylum_taxa_name	phylum_taxa_count	class_taxa_name	class_taxa_count	\
0	Proteobacteria	521947	Gammaproteobacteria	474642	
1	Actinobacteriota	256165	Actinobacteria	255633	
2	Firmicutes	224467	Bacilli	224467	
3	Bacteroidota	13018	Alphaproteobacteria	47278	
4	Firmicutes_A	11780	Bacteroidia	12755	

	order_taxa_name	order_taxa_count	family_taxa_name	family_taxa_count	\
0	Enterobacterales	196062	Enterobacteriaceae	165898	
1	Mycobacteriales	179848	Mycobacteriaceae	164260	
2	Pseudomonadales	158378	Pseudomonadaceae	118544	
3	Burkholderiales	99779	Burkholderiaceae	84793	
4	Lactobacillales	93623	Streptococcaceae	74771	

	genus_taxa_name	genus_taxa_count	species_taxa_name	\
0	Mycobacterium	104666	Mycobacterium tuberculosis	
1	Pseudomonas	74864	Pseudomonas aeruginosa	
2	Staphylococcus	73841	Staphylococcus aureus	
3	Streptococcus	73532		
4	Escherichia	67960	Streptococcus pneumoniae	

	species_taxa_count
0	93094

1	74092
2	63573
3	52225
4	50516

```
[15]: taxonomy_resolution_6.head()
```

```
[15]:
```

	phylum_taxa_name	phylum_taxa_count	class_taxa_name	class_taxa_count	\
0	Proteobacteria	521947	Gammaproteobacteria	474642	
1	Actinobacteriota	256165	Actinobacteria	255633	
2	Firmicutes	224467	Bacilli	224467	
3	Bacteroidota	13018	Alphaproteobacteria	47278	
4	Firmicutes_A	11780	Bacteroidia	12755	

	order_taxa_name	order_taxa_count	family_taxa_name	family_taxa_count	\
0	Enterobacterales	196062	Enterobacteriaceae	165898	
1	Mycobacteriales	179848	Mycobacteriaceae	164260	
2	Pseudomonadales	158378	Pseudomonadaceae	118544	
3	Burkholderiales	99779	Burkholderiaceae	84793	
4	Lactobacillales	93623	Streptococcaceae	74771	

	genus_taxa_name	genus_taxa_count	species_taxa_name	\
0	Mycobacterium	104666	Mycobacterium tuberculosis	
1	Pseudomonas	74864	Pseudomonas aeruginosa	
2	Staphylococcus	73841	Staphylococcus aureus	
3	Streptococcus	73532		
4	Escherichia	67960	Streptococcus pneumoniae	

	species_taxa_count
0	93094
1	74092
2	63573
3	52225
4	50516

```
[16]: taxonomy_resolution_7.head()
```

```
[16]:
```

	phylum_taxa_name	phylum_taxa_count	class_taxa_name	class_taxa_count	\
0	Proteobacteria	521947	Gammaproteobacteria	474642	
1	Actinobacteriota	256165	Actinobacteria	255633	
2	Firmicutes	224467	Bacilli	224467	
3	Bacteroidota	13018	Alphaproteobacteria	47278	
4	Firmicutes_A	11780	Bacteroidia	12755	

	order_taxa_name	order_taxa_count	family_taxa_name	family_taxa_count	\
0	Enterobacterales	196062	Enterobacteriaceae	165898	
1	Mycobacteriales	179848	Mycobacteriaceae	164260	

2	Pseudomonadales	158378	Pseudomonadaceae	118544
3	Burkholderiales	99779	Burkholderiaceae	84793
4	Lactobacillales	93623	Streptococcaceae	74771

	genus_taxa_name	genus_taxa_count	species_taxa_name \
0	Mycobacterium	104666	Mycobacterium tuberculosis
1	Pseudomonas	74864	Pseudomonas aeruginosa
2	Staphylococcus	73841	Staphylococcus aureus
3	Streptococcus	73532	
4	Escherichia	67960	Streptococcus pneumoniae

	species_taxa_count
0	93094
1	74092
2	63573
3	52225
4	50516

As it can be seen in the above outputs when dealing with MAG's it is likely to get low taxonomic resolution resolution.

The first dataset has RefSeq + MAG datasets, others have just RefSeq sequences. This influences the taxonomic resolution.

In species level resolution merged dataset has 115,269.00 missing information in the other dataset it is just 52,225.00.

```
[17]: def get_unique_gcf_count(taxonomy_file: pd.DataFrame, gcfs: pd.DataFrame) -> pd.
      DataFrame:
          """
          Calculates the unique GCF count for each genus in the taxonomy file and
          returns a sorted DataFrame.

          Args:
              taxonomy_file (pd.DataFrame): DataFrame containing taxonomy information
              with genus names at index 5.
              gcfs (pd.DataFrame): DataFrame containing GCF IDs in a column named
              "gcf_id".

          Returns:
              pd.DataFrame: Sorted DataFrame with columns:
                  - Genus_Taxonomy: Genus names
                  - gcf_count_unique: Number of unique GCF IDs per genus
                  - gcf_index: List of unique GCF IDs
                  - gcf_count: Corresponding counts of unique GCF IDs
          """

          # Extract unique genus names
```

```

genus_resolution = taxonomy_file.iloc[:, 5].value_counts().index

# Create a results DataFrame
results_df = pd.DataFrame(
    index=genus_resolution,
    columns=["gcf_count_unique", "gcf_index", "gcf_count"],
)

# Iterate through each genus and fill DataFrame
for genus in genus_resolution:
    genus_gcfs = gcfs.loc[taxonomy_file[taxonomy_file[5] == genus].index,
↳ "gcf_id"]
    unique_gcfs, counts = np.unique(genus_gcfs, return_counts=True)
    results_df.loc[genus] = [len(unique_gcfs), list(unique_gcfs),
↳ list(counts)]

# Save results to descriptively named CSV files
results_df.to_csv(
    f"reproducibility_results_{taxonomy_file.name}.csv",
    index_label="Genus_Taxonomy",
)

# Sort by unique GCF count in descending order
results_df = results_df.sort_values("gcf_count_unique", ascending=False)

results_df.to_csv(
    f"reproducibility_results_{taxonomy_file.name}_sorted.csv",
    index_label="Genus_Taxonomy",
)

return results_df

```

```

[18]: # Calculate unique GCF counts for each BiG-SLiCE threshold (4-7)
taxonomy_4_results: pd.DataFrame = get_unique_gcf_count(taxonomy_file =
↳ taxonomy_4, gcfs = thresholds_4_gcfs)

taxonomy_5_results: pd.DataFrame = get_unique_gcf_count(taxonomy_file =
↳ taxonomy_5, gcfs = thresholds_5_gcfs)

taxonomy_6_results: pd.DataFrame = get_unique_gcf_count(taxonomy_file =
↳ taxonomy_6, gcfs = thresholds_6_gcfs)

taxonomy_7_results: pd.DataFrame = get_unique_gcf_count(taxonomy_file =
↳ taxonomy_7, gcfs = thresholds_7_gcfs)

```

```

[19]: print(taxonomy_4_results.head())

```

```

gcf_count_unique \
5
Streptomyces      10820
Pseudomonas_E     8703
Nocardia          1517
Micromonospora    1421

```

```

gcf_index \
5
Streptomyces      [0, 1, 3, 7, 21, 24, 29, 30, 35, 37, 39, 40, 4...
Pseudomonas_E    [0, 7, 30, 41, 42, 44, 45, 46, 50, 53, 56, 57,...
Nocardia          [0, 5, 41, 44, 46, 49, 59, 79, 86, 153, 154, 1...
Micromonospora    [0, 28, 46, 50, 76, 118, 837, 855, 883, 1061, ...

```

```

gcf_count
5
Streptomyces      [1254, 3, 2, 5, 1, 8, 29, 16, 21, 17, 4, 2, 20...
Pseudomonas_E    [186, 1, 21, 1, 4, 6, 1, 72, 10, 4, 1, 4, 1, 1...
Nocardia          [36, 1, 122, 3, 1, 1, 1, 10, 1, 1, 7, 1, 41, 4...
Micromonospora    [25, 1, 12, 1, 1, 4, 1, 1, 2, 12, 1, 2, 1, 1, ...

```

```
[20]: print(taxonomy_5_results.head())
```

```

gcf_count_unique \
5
Streptomyces      6798
Nocardia          3971
Pseudomonas_E     1146
Amycolatopsis     1101

```

```

gcf_index \
5
Streptomyces      [20, 24, 25, 31, 35, 36, 38, 44, 49, 51, 60, 6...
Nocardia          [0, 7, 23, 24, 25, 28, 29, 31, 33, 34, 35, 36,...
Pseudomonas_E    [22, 53, 60, 69, 106, 121, 249, 298, 561, 584,...
Amycolatopsis     [24, 36, 38, 40, 47, 65, 68, 96, 106, 115, 121...

```

```

gcf_count
5
Streptomyces      [1, 34, 4, 1, 6, 1, 7, 1, 3, 22, 1, 1, 7, 5, 5...
Nocardia          [1, 1, 9, 14, 1, 2, 1, 24, 4, 2, 1, 46, 12, 2,...
Pseudomonas_E    [1, 2, 1, 1, 2, 13, 1, 4, 8, 2, 13, 1, 2, 1, 1...

```

```
Amycolatopsis [1, 1, 2, 2, 3, 1, 1, 1, 2, 10, 6, 1, 10, 1, 3...
```

```
[21]: print(taxonomy_6_results.head())
```

```

gcf_count_unique \
5
Streptomyces      5136
                  3110
Nocardia          890
Pseudomonas_E     881
Amycolatopsis     700

gcf_index \
5
Streptomyces [2, 3, 23, 25, 28, 30, 32, 34, 35, 36, 38, 43,...
              [0, 3, 4, 8, 10, 23, 25, 26, 27, 28, 30, 31, 3...
Nocardia     [3, 32, 40, 50, 162, 216, 323, 361, 362, 368, ...
Pseudomonas_E [28, 30, 31, 32, 35, 47, 62, 70, 78, 110, 130,...
Amycolatopsis [25, 30, 32, 55, 62, 216, 263, 309, 341, 354, ...

gcf_count
5
Streptomyces [9, 5, 1, 31, 14, 108, 93, 2, 4, 7, 28, 5, 1, ...
              [1, 1, 12, 1, 6, 8, 43, 4, 10, 82, 47, 20, 20,...
Nocardia     [1, 14, 2, 1, 10, 3, 2, 1, 1, 2, 1, 1, 12, 1, ...
Pseudomonas_E [227, 89, 3, 3, 40, 10, 23, 11, 4, 1, 5, 310, ...
Amycolatopsis [10, 2, 5, 2, 2, 2, 12, 1, 5, 1, 2, 3, 1, 1, 1...
```

```
[22]: print(taxonomy_7_results.head())
```

```

gcf_count_unique \
5
Streptomyces      3363
                  2253
Pseudomonas_E     694
Nocardia          669
Amycolatopsis     487

gcf_index \
5
Streptomyces [1, 2, 3, 6, 7, 10, 16, 20, 21, 23, 27, 30, 37...
              [0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 14, 15, 16, 17...
Pseudomonas_E [1, 4, 7, 8, 10, 14, 16, 27, 31, 36, 42, 49, 5...
Nocardia      [1, 3, 10, 37, 46, 50, 59, 90, 98, 102, 104, 1...
Amycolatopsis [1, 6, 7, 10, 46, 59, 66, 80, 98, 121, 124, 12...

gcf_count
5
Streptomyces [166, 1, 5, 37, 125, 30, 7, 7, 1, 576, 9, 2, 9...
```

	[1, 46, 12, 1, 17, 44, 54, 11, 4, 153, 31, 5, ...
Pseudomonas_E	[2, 1, 89, 1, 642, 11, 40, 3, 10, 55, 1, 11, 8...
Nocardia	[18, 1, 2, 1, 1, 1, 6, 11, 1, 3, 1, 3, 3, 1, 1...
Amycolatopsis	[6, 14, 3, 1, 4, 2, 1, 14, 6, 1, 1, 9, 65, 1, ...

2 END OF ADDITIONAL EFFORTS

```
[23]: # thresholds_4_gcfs: np.ndarray = np.array(STable2.gcf_id.unique())
# thresholds_5_gcfs: np.ndarray = np.array(STable3.gcf_id.unique())
# thresholds_6_gcfs: np.ndarray = np.array(STable4.gcf_id.unique())
# thresholds_7_gcfs: np.ndarray = np.array(STable5.gcf_id.unique())
# bgc_ids: DataFrame = STable1.bgc_ids.str.split(",", expand=True)
# bgc_ids = (bgc_ids.apply(pd.to_numeric, downcast="unsigned"))
# bgc_ids[["dataset_name", "AccNo", "taxonomy"]] = STable1[["dataset_name",
↳ "AccNo", "taxonomy"]]
# STable2["bgc_id"] = STable2["bgc_id"].astype(np.int64)
# STable2.index = STable2["bgc_id"]
# STable2["taxonomy"] = ""
# from pandarallel import pandarallel
# import os
# pandarallel.initialize(nb_workers=os.cpu_count())
# def add_taxonomy_to_table(gcfs_array: np.ndarray, stable: DataFrame) -> None:
#     # print(type(gcfs_array))
#     # print(gcfs_array[5])
#     for gcf in gcfs_array:
#         # print(f"gcf{gcf}\n")
#         bgc_ids_of_gcf: np.ndarray = np.array(STable2[STable2.gcf_id ==
↳ gcf]["bgc_id"])
#         for bgc_id in bgc_ids_of_gcf:
#             # print(f"bgc_id{bgc_id}\n")
#             # print(f"type(bgc_id{type(bgc_id)}\n")
#             bgc_tax = bgc_ids[bgc_ids.eq(bgc_id).
↳ any(axis="columns")]["taxonomy"].iloc[0]
#             stable.loc[bgc_id, "taxonomy"] = bgc_tax
# add_taxonomy_to_table = np.vectorize(add_taxonomy_to_table)

# add_taxonomy_to_table(gcfs_array = thresholds_4_gcfs[-10:], stable = STable2)
```