RBP-ID

Development of a Web-Based Bioinformatics Tool for the Identification of Phage Receptor-Binding Proteins

Presented by Cátia Rosário
Oriented by Silvio Santos and Óscar Dias

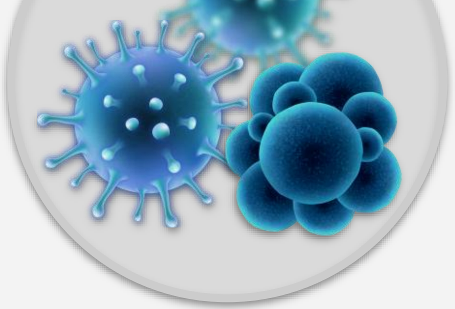
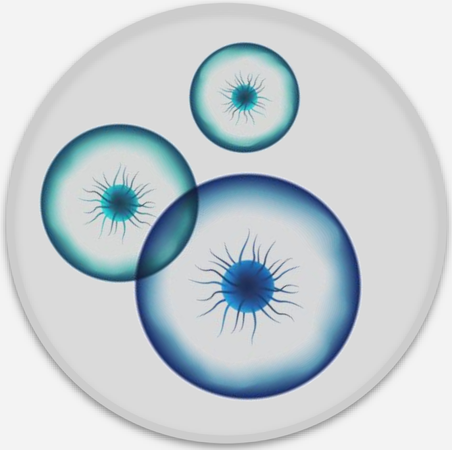
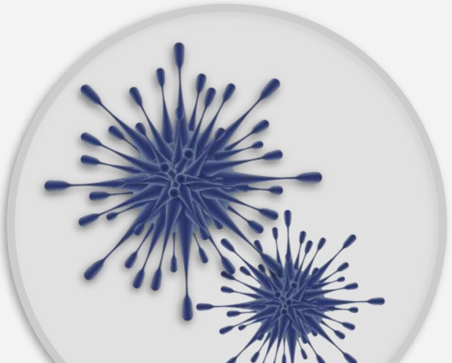
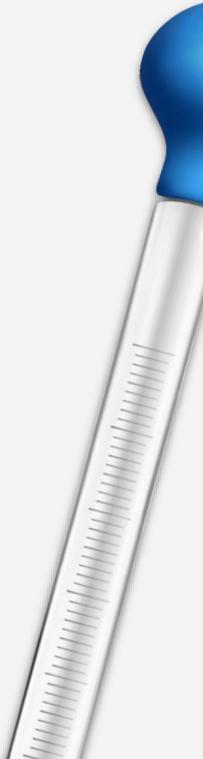
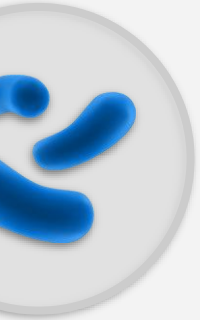
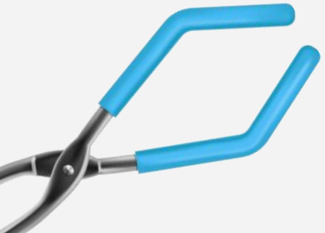
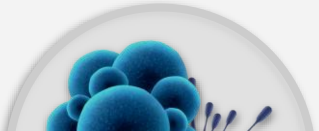





Table of contents

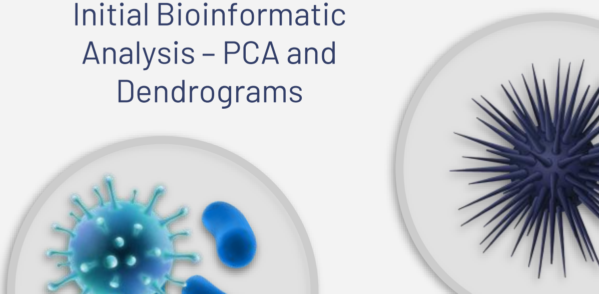




01

Milestones

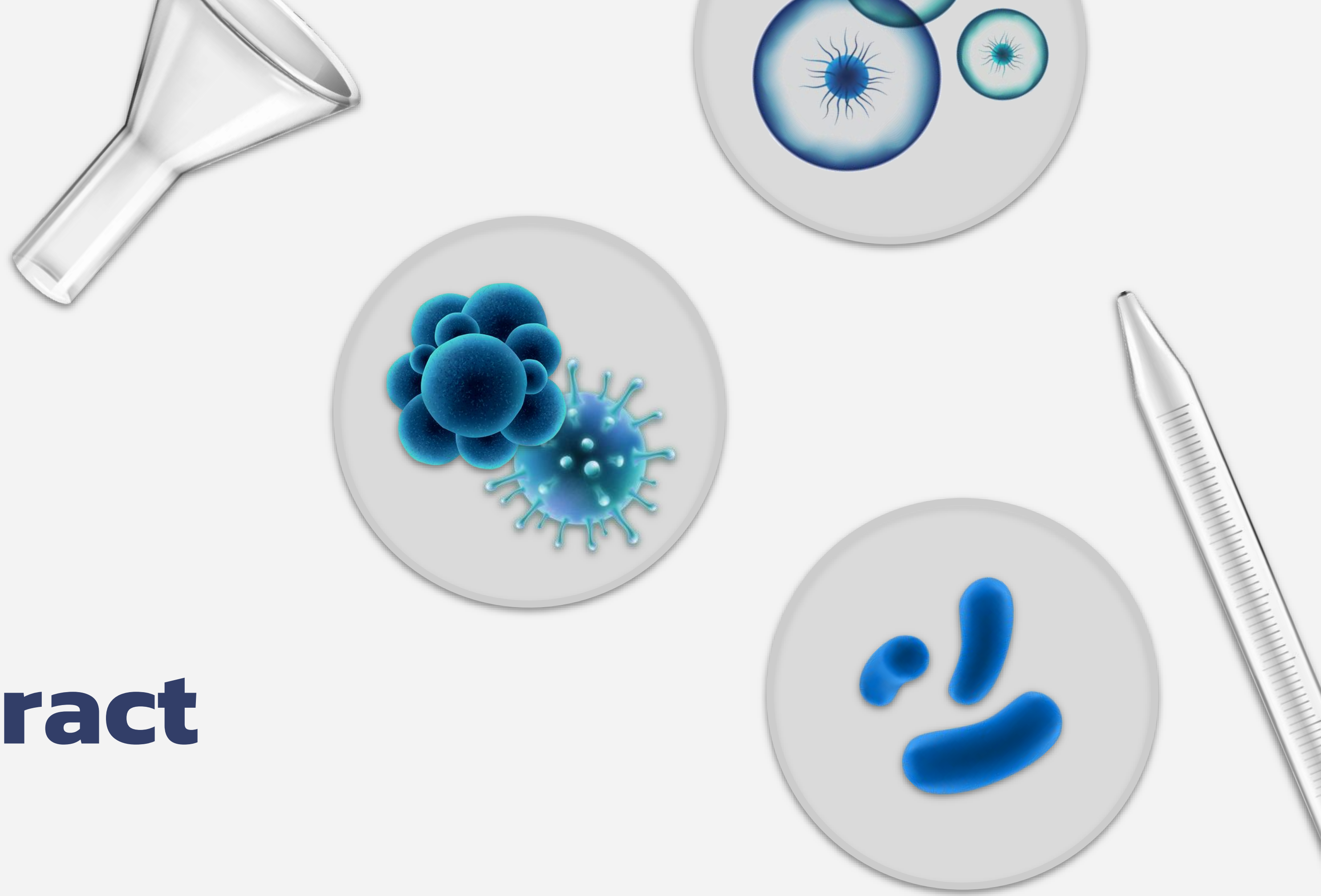
Milestones reached

- Initial Research about the theme
- Literature Compilation about Viruses and RBPs
- 1st Dataset creation with manually compiled RBPs candidates
- RBPs Validation through published experimental data
- 2nd Dataset creation with computationally predicted RBPs from a compilation of Datasets
- 3nd Dataset creation with computationally predicted and validated RBPs
- Initial Bioinformatic Analysis – Clustering with cd-hit and K-means
- Initial Bioinformatic Analysis – PCA and Dendrograms



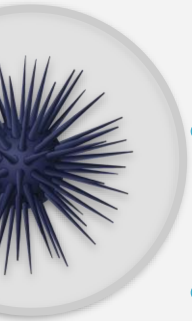
02

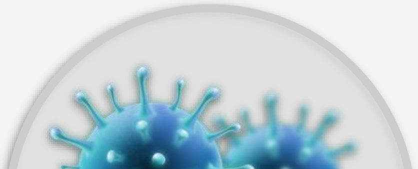
Abstract



Abstract

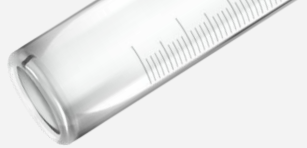
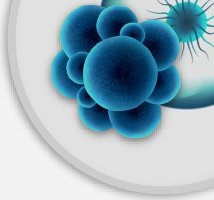
- **Bacteriophages:** Viruses that infect bacteria with key roles in biotechnology and medicine.
- **Receptor-Binding Proteins (RBPs):** Determine host specificity by mediating bacterial recognition and attachment.
- **Challenge:** RBPs exhibit high sequence diversity and low conservation, making identification difficult.
- **Solution – RBP-ID:** A web-based bioinformatics tool for systematic RBP prediction in phage genomes.





03

Introduction

Introduction

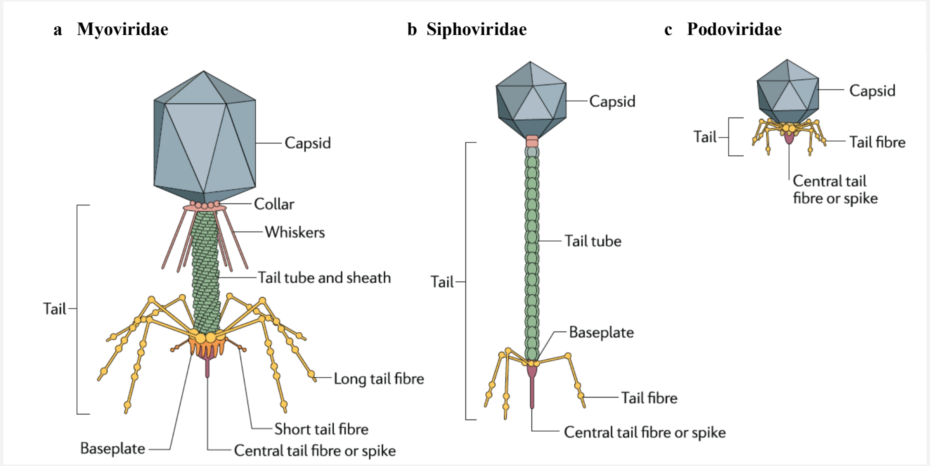
Background

Phage display & delivery systems: Technologies for gene editing and therapeutic biomolecule delivery.
Adaptation mechanisms: Co-evolution with bacteria through receptor mutations and CRISPR-Cas defence systems.
Computational tools for phage genome annotation:

- Hybrid approaches: HMM searches and ML classifiers
- Host prediction via RBP analysis: Random Forest and SVMs, XGBoost
- PHYPred: Functional annotation tools for phage-encoded enzymes
- PhANNs

Importance

Biotechnological & therapeutic: Antibiotic resistance solutions, microbiome engineering, biosensing and precision medicine
Phages vs. antibiotics: Target specific pathogens without harming beneficial bacteria, a key advantage in antimicrobial resistance (AMR)
Synthetic biology advances: Engineered phages and phage-derived enzymes broaden applications in medicine, agriculture, and industry
Need for advanced bioinformatics tools

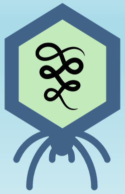


Morphotypes

Tail Fibre World

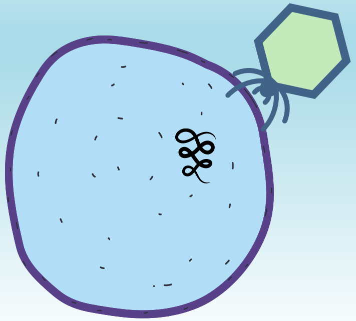
Short, Long, Spikey, Contractile, Non-Contractile

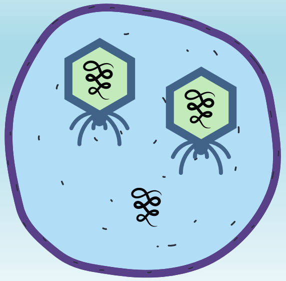
HOW DO VIRUSES REPLICATE



The virus invades the host cell

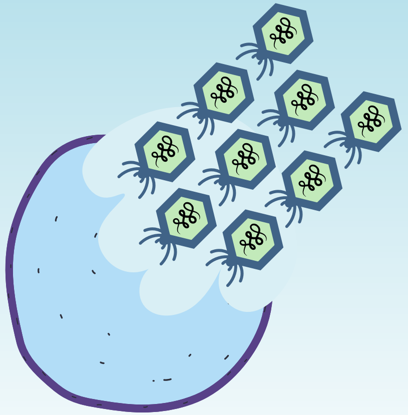
The virus injects its genetic material into the host cell





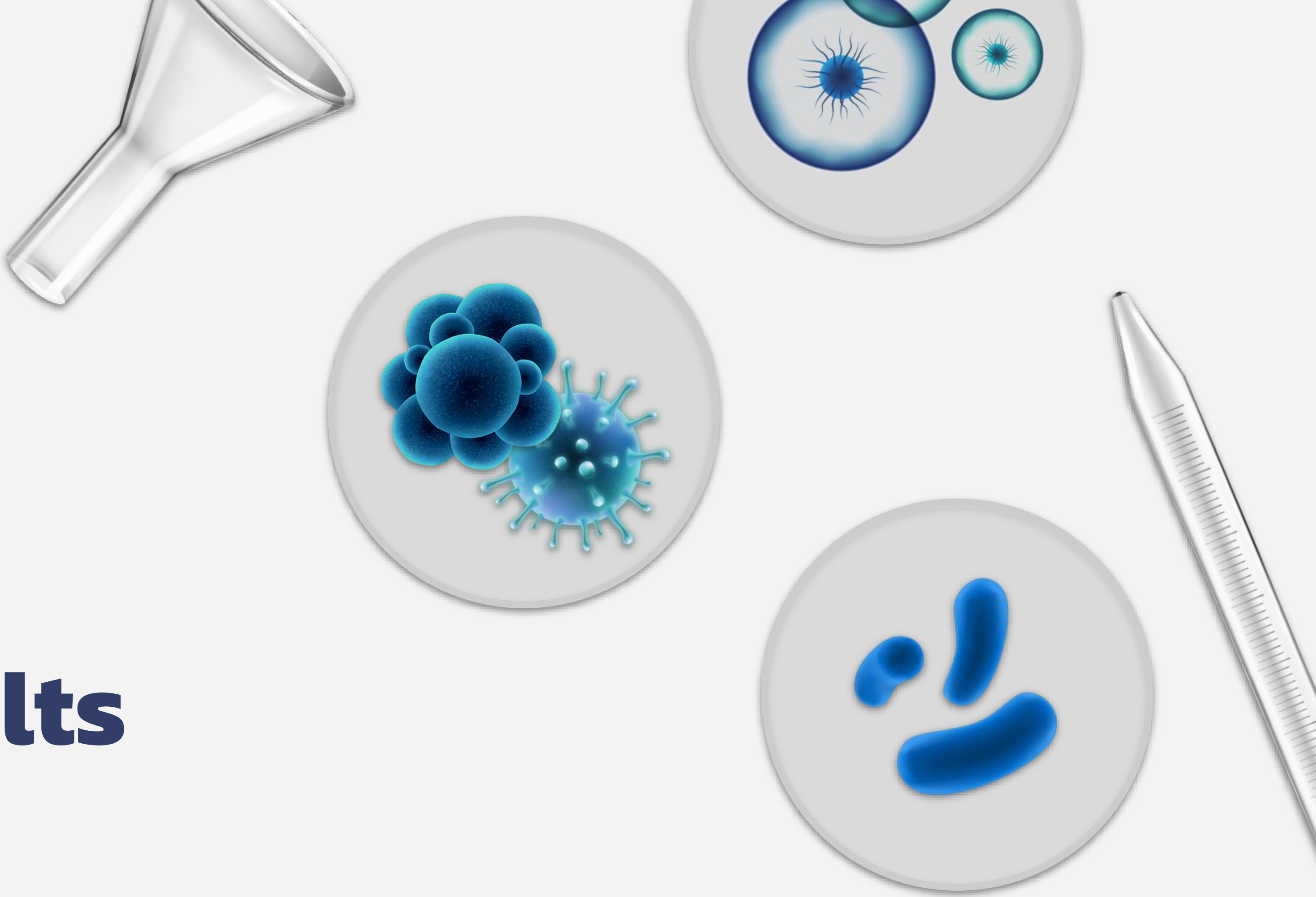
The virus's genetic material reprograms the host's DNA and starts to replicate itself

When the host cell is filled with virus copies, it bursts, releasing them into the bloodstream to repeat the process



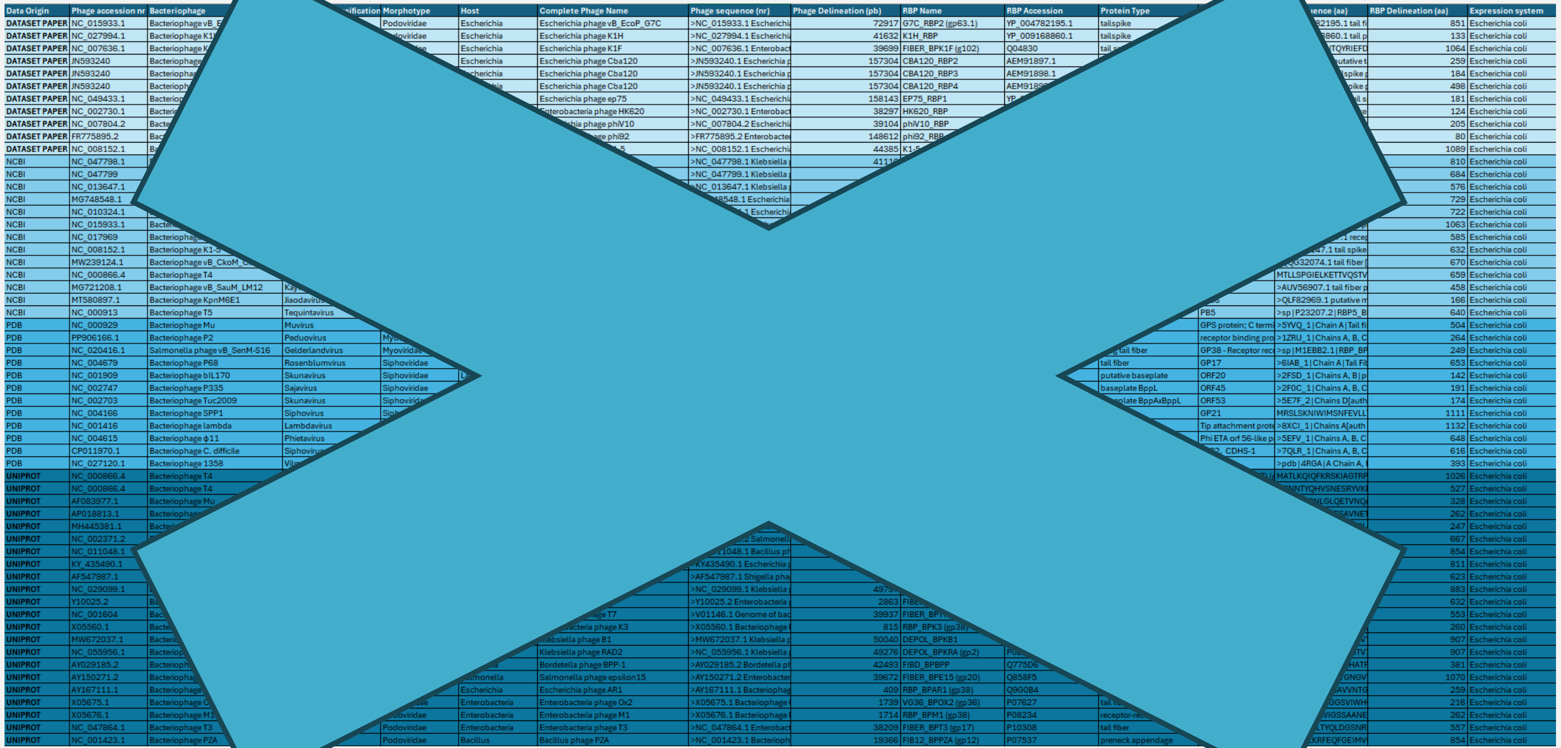
04

Results



4.1

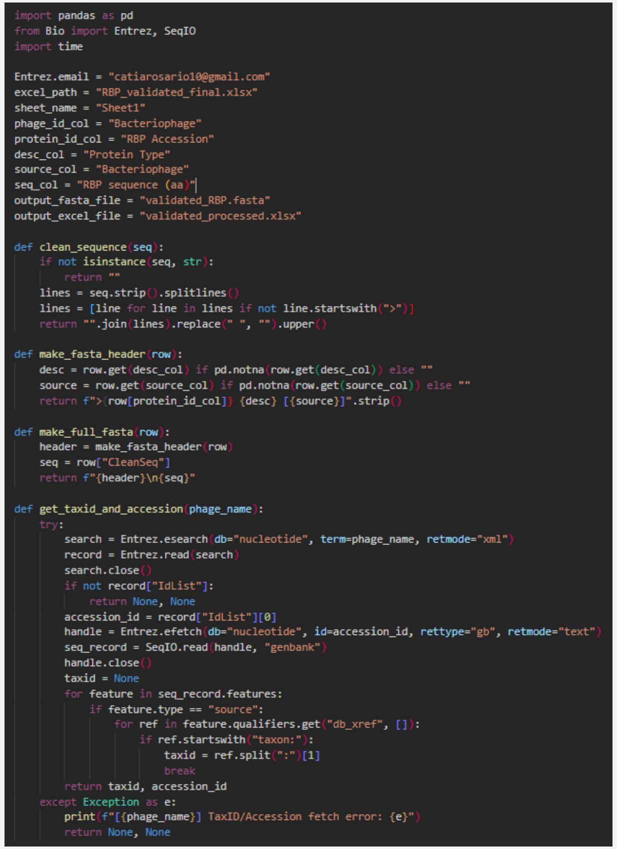
1st Dataset



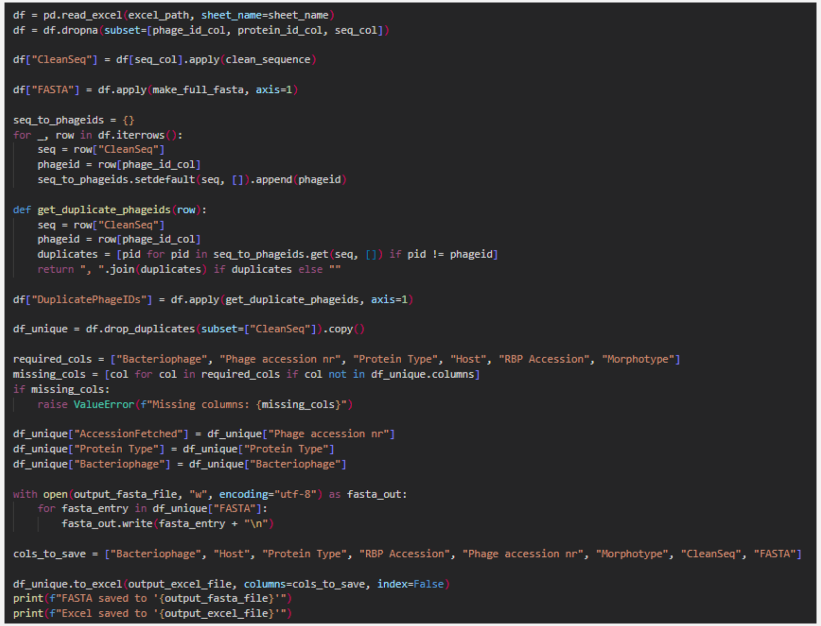
Data Origin	Phage accession nr	Bacteriophage	sification	Morphotype	Host	Complete Phage Name	Phage sequence (nr)	Phage Delineation (pb)	RBP Name	RBP Accession	Protein Type		ence (aa)	RBP Delineation (aa)	Expression system
DATASET PAPER	NC_015933.1	Bacteriophage vB_E		Podoviridae	Escherichia	Escherichia phage vB_EcoP_G7C	>NC_015933.1 Escherichia	72917	G7C_RBP2 (gp63.1)	YP_004782195.1	tailspike		82195.1 tail fi	851	Escherichia coli
DATASET PAPER	NC_027994.1	Bacteriophage K1		doviridae	Escherichia	Escherichia phage K1H	>NC_027994.1 Escherichia	41632	K1H_RBP	YP_009168860.1	tailspike		8860.1 tail p	133	Escherichia coli
DATASET PAPER	NC_007636.1	Bacteriophage K			Escherichia	Escherichia phage K1F	>NC_007636.1 Enterobact	39699	FIBER_BPK1F (g102)	Q04830	tail s		ITQYRIEFD	1064	Escherichia coli
DATASET PAPER	JN593240	Bacteriophage			Escherichia	Escherichia phage Cba120	>JN593240.1 Escherichia p	157304	CBA120_RBP2	AEM91897.1			utative t	259	Escherichia coli
DATASET PAPER	JN593240	Bacteriophag			cherichia	Escherichia phage Cba120	>JN593240.1 Escherichia p	157304	CBA120_RBP3	AEM91898.1			lspike p	184	Escherichia coli
DATASET PAPER	JN593240	Bacterioph			hia	Escherichia phage Cba120	>JN593240.1 Escherichia p	157304	CBA120_RBP4	AEM9189			pike p	498	Escherichia coli
DATASET PAPER	NC_049433.1	Bacteriop				Escherichia phage ep75	>NC_049433.1 Escherichia	158143	EP75_RBP1	YP_			il s	181	Escherichia coli
DATASET PAPER	NC_002730.1	Bacteri				nterobacteria phage HK620	>NC_002730.1 Enterobact	38297	HK620_RBP					124	Escherichia coli
DATASET PAPER	NC_007804.2	Bacte				hia phage phiV10	>NC_007804.2 Escherichia	39104	phiV10_RBP					205	Escherichia coli
DATASET PAPER	FR775895.2	Bac				ge phi92	>FR775895.2 Enterobacter	148612	phi92_RBP					80	Escherichia coli
DATASET PAPER	NC_008152.1	Ba				1-5	>NC_008152.1 Escherichia	44385	K1-5					1089	Escherichia coli
NCBI	NC_047798.1						>NC_047798.1 Klebsiella	4111						810	Escherichia coli
NCBI	NC_047799						>NC_047799.1 Klebsiella							684	Escherichia coli
NCBI	NC_013647.1						>NC_013647.1 Klebsiella							576	Escherichia coli
NCBI	MG748548.1						48548.1 Escherichia							729	Escherichia coli
NCBI	NC_010324.1						.1 Escherichia							722	Escherichia coli
NCBI	NC_015933.1	Bacte												1063	Escherichia coli
NCBI	NC_017969	Bacteriophag											.1 recep	585	Escherichia coli
NCBI	NC_008152.1	Bacteriophage K1-5											47.1 tail spike	632	Escherichia coli
NCBI	MW239124.1	Bacteriophage vB_CkoM_C											QG32074.1 tail fiber [	670	Escherichia coli
NCBI	NC_000866.4	Bacteriophage T4											MTLLSPGIELKETTVQSTV	659	Escherichia coli
NCBI	MG721208.1	Bacteriophage vB_SauM_LM12	Kay										>AUV56907.1 tail fiber p	458	Escherichia coli
NCBI	MT580897.1	Bacteriophage KpnM6E1	Jiaodavirus										>QLF82969.1 putative m	166	Escherichia coli
NCBI	NC_000913	Bacteriophage T5	Tequintavirus									PB5	>sp|P23207.2|RBP5_B	640	Escherichia coli
PDB	NC_000929	Bacteriophage Mu	Muvirus									GPS protein; C termi	>5YVQ_1|Chain A|Tail fi	504	Escherichia coli
PDB	PP906166.1	Bacteriophage P2	Peduovirus	Myo								receptor binding pro	>1ZRU_1|Chains A, B, C	264	Escherichia coli
PDB	NC_020416.1	Salmonella phage vB_SenM-S16	Gelderlandvirus	Myoviridae							g tail fiber	GP38 - Receptor rec	>sp|M1EBB2.1|RBP_BP	249	Escherichia coli
PDB	NC_004679	Bacteriophage P68	Rosenblumvirus	Siphoviridae							tail fiber	GP17	>6IAB_1|Chain A|Tail Fi	653	Escherichia coli
PDB	NC_001909	Bacteriophage bIL170	Skunavirus	Siphoviridae	La						putative baseplate	ORF20	>2FSD_1|Chains A, B|p	142	Escherichia coli
PDB	NC_002747	Bacteriophage P335	Sajavirus	Siphoviridae							baseplate BppL	ORF45	>2F0C_1|Chains A, B, C	191	Escherichia coli
PDB	NC_002703	Bacteriophage Tuc2009	Skunavirus	Siphovirida							eplate BppAxBppL	ORF53	>5E7F_2|Chains D[auth	174	Escherichia coli
PDB	NC_004166	Bacteriophage SPP1	Siphovirus	Siph								GP21	MRSLSKNIWIMSNFEVLL	1111	Escherichia coli
PDB	NC_001416	Bacteriophage lambda	Lambdavirus									Tip attachment prote	>8XCI_1|Chains A[auth	1132	Escherichia coli
PDB	NC_004615	Bacteriophage ϕ11	Phietavirus									Phi ETA orf 56-like p	>5EFV_1|Chains A, B, C	648	Escherichia coli
PDB	CP011970.1	Bacteriophage C. difficile	Siphovirus									2, CDHS-1	>7QLR_1|Chains A, B, C	616	Escherichia coli
PDB	NC_027120.1	Bacteriophage 1358	Vil										>pdb|4RGA|A Chain A,	393	Escherichia coli
UNIPROT	NC_000866.4	Bacteriophage T4											MATLKQIQFKRSKIAGTRP	1026	Escherichia coli
UNIPROT	NC_000866.4	Bacteriophage T4											SNNTYQHVSNESRYVK	527	Escherichia coli
UNIPROT	AF083977.1	Bacteriophage Mu											NLGLQETVNQ	328	Escherichia coli
UNIPROT	AP018813.1	Bacteriophag											ESAVNET	262	Escherichia coli
UNIPROT	MH445381.1	Bacteri												247	Escherichia coli
UNIPROT	NC_002371.2						.2 Salmonell							667	Escherichia coli
UNIPROT	NC_011048.1						11048.1 Bacillus ph							854	Escherichia coli
UNIPROT	KY_435490.1						>KY435490.1 Escherichia p							811	Escherichia coli
UNIPROT	AF547987.1						>AF547987.1 Shigella pha							623	Escherichia coli
UNIPROT	NC_029099.1						>NC_029099.1 Klebsiella	4979						883	Escherichia coli
UNIPROT	Y10025.2	Ba					>Y10025.2 Enterobacteria	2863	FIBE					632	Escherichia coli
UNIPROT	NC_001604	Bac				age T7	>V01146.1 Genome of bac	39937	FIBER_BP					553	Escherichia coli
UNIPROT	X05560.1	Bacte				acteria phage K3	>X05560.1 Bacteriophage	815	RBP_BPK3 (gp38)					260	Escherichia coli
UNIPROT	MW672037.1	Bacteri				lebsiella phage B1	>MW672037.1 Klebsiella p	50040	DEPOL_BPKB1				V	907	Escherichia coli
UNIPROT	NC_055956.1	Bacteriop				Klebsiella phage RAD2	>NC_055956.1 Klebsiella	49276	DEPOL_BPKRA (gp2)	P0			STVT	907	Escherichia coli
UNIPROT	AY029185.2	Bacterioph			a	Bordetella phage BPP-1	>AY029185.2 Bordetella ph	42493	FIBD_BPBPP	Q775D6			HATF	381	Escherichia coli
UNIPROT	AY150271.2	Bacteriophag			almonella	Salmonella phage epsilon15	>AY150271.2 Enterobacter	39672	FIBER_BPE15 (gp20)	Q858F5			GNGV	1070	Escherichia coli
UNIPROT	AY167111.1	Bacteriophage			Escherichia	Escherichia phage AR1	>AY167111.1 Bacteriophag	409	RBP_BPAR1 (gp38)	Q9G0B4			AVVNTG	259	Escherichia coli
UNIPROT	X05675.1	Bacteriophage O		dae	Enterobacteria	Enterobacteria phage Ox2	>X05675.1 Bacteriophage	1739	VG36_BPOX2 (gp36)	P07627	tail fi		GGSVIWH	216	Escherichia coli
UNIPROT	X05676.1	Bacteriophage M1		odoviridae	Enterobacteria	Enterobacteria phage M1	>X05676.1 Bacteriophage	1714	RBP_BPM1 (gp38)	P08234	receptor-re		WIGSSAANE	262	Escherichia coli
UNIPROT	NC_047864.1	Bacteriophage T3		Podoviridae	Enterobacteria	Enterobacteria phage T3	>NC_047864.1 Enterobact	38209	FIBER_BPT3 (gp17)	P10308	tail fiber		LTYQLDGSNR	557	Escherichia coli
UNIPROT	NC_001423.1	Bacteriophage PZA		Podoviridae	Bacillus	Bacillus phage PZA	>NC_001423.1 Bacterioph	19366	FIB12_BPPZA (gp12)	P07537	preneck appendage		KRFEQFGEIMV	854	Escherichia coli

1st Dataset – Validated RBPs

Validated RBPs – Processement with Python

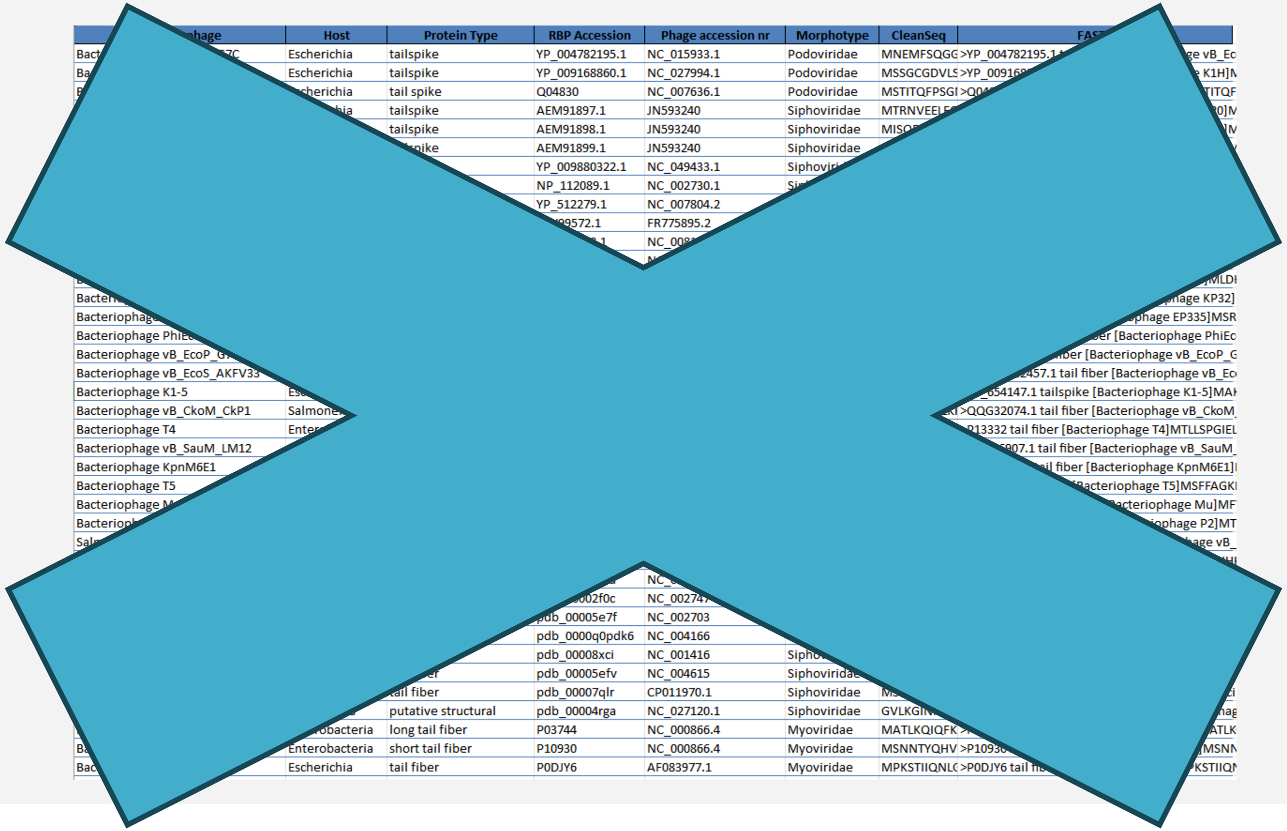


```python
import pandas as pd
from Bio import Entrez, SeqIO
import time

Entrez.email = "catiarosario10@gmail.com"
excel_path = "RBP_validated_final.xlsx"
sheet_name = "Sheet1"
phage_id_col = "Bacteriophage"
protein_id_col = "RBP Accession"
desc_col = "Protein Type"
source_col = "Bacteriophage"
seq_col = "RBP sequence (aa)"
output_fasta_file = "validated_RBP.fasta"
output_excel_file = "validated_processed.xlsx"

def clean_sequence(seq):
    if not isinstance(seq, str):
        return ""
    lines = seq.strip().splitlines()
    lines = [line for line in lines if not line.startswith(">")]
    return "".join(lines).replace(" ", "").upper()

def make_fasta_header(row):
    desc = row.get(desc_col) if pd.notna(row.get(desc_col)) else ""
    source = row.get(source_col) if pd.notna(row.get(source_col)) else ""
    return f">{row[protein_id_col]} {desc} [{source}]".strip()

def make_full_fasta(row):
    header = make_fasta_header(row)
    seq = row["CleanSeq"]
    return f"{header}\n{seq}"

def get_taxid_and_accession(phage_name):
    try:
        search = Entrez.esearch(db="nucleotide", term=phage_name, retmode="xml")
        record = Entrez.read(search)
        search.close()
        if not record["IdList"]:
            return None, None
        accession_id = record["IdList"][0]
        handle = Entrez.efetch(db="nucleotide", id=accession_id, rettype="gb", retmode="text")
        seq_record = SeqIO.read(handle, "genbank")
        handle.close()
        taxid = None
        for feature in seq_record.features:
            if feature.type == "source":
                for ref in feature.qualifiers.get("db_xref", []):
                    if ref.startswith("taxon:"):
                        taxid = ref.split(":")[1]
                        break
        return taxid, accession_id
    except Exception as e:
        print(f"[{phage_name}] TaxID/Accession fetch error: {e}")
        return None, None
```



```python
df = pd.read_excel(excel_path, sheet_name=sheet_name)
df = df.dropna(subset=[phage_id_col, protein_id_col, seq_col])

df["CleanSeq"] = df[seq_col].apply(clean_sequence)

df["FASTA"] = df.apply(make_full_fasta, axis=1)

seq_to_phageids = {}
for _, row in df.iterrows():
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    seq_to_phageids.setdefault(seq, []).append(phageid)

def get_duplicate_phageids(row):
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    duplicates = [pid for pid in seq_to_phageids.get(seq, []) if pid != phageid]
    return ", ".join(duplicates) if duplicates else ""

df["DuplicatePhageIDs"] = df.apply(get_duplicate_phageids, axis=1)

df_unique = df.drop_duplicates(subset=["CleanSeq"]).copy()

required_cols = ["Bacteriophage", "Phage accession nr", "Protein Type", "Host", "RBP Accession", "Morphotype"]
missing_cols = [col for col in required_cols if col not in df_unique.columns]
if missing_cols:
    raise ValueError(f"Missing columns: {missing_cols}")

df_unique["AccessionFetched"] = df_unique["Phage accession nr"]
df_unique["Protein Type"] = df_unique["Protein Type"]
df_unique["Bacteriophage"] = df_unique["Bacteriophage"]

with open(output_fasta_file, "w", encoding="utf-8") as fasta_out:
    for fasta_entry in df_unique["FASTA"]:
        fasta_out.write(fasta_entry + "\n")

cols_to_save = ["Bacteriophage", "Host", "Protein Type", "RBP Accession", "Phage accession nr", "Morphotype", "CleanSeq", "FASTA"]

df_unique.to_excel(output_excel_file, columns=cols_to_save, index=False)
print(f"FASTA saved to '{output_fasta_file}'")
print(f"Excel saved to '{output_excel_file}'")
```

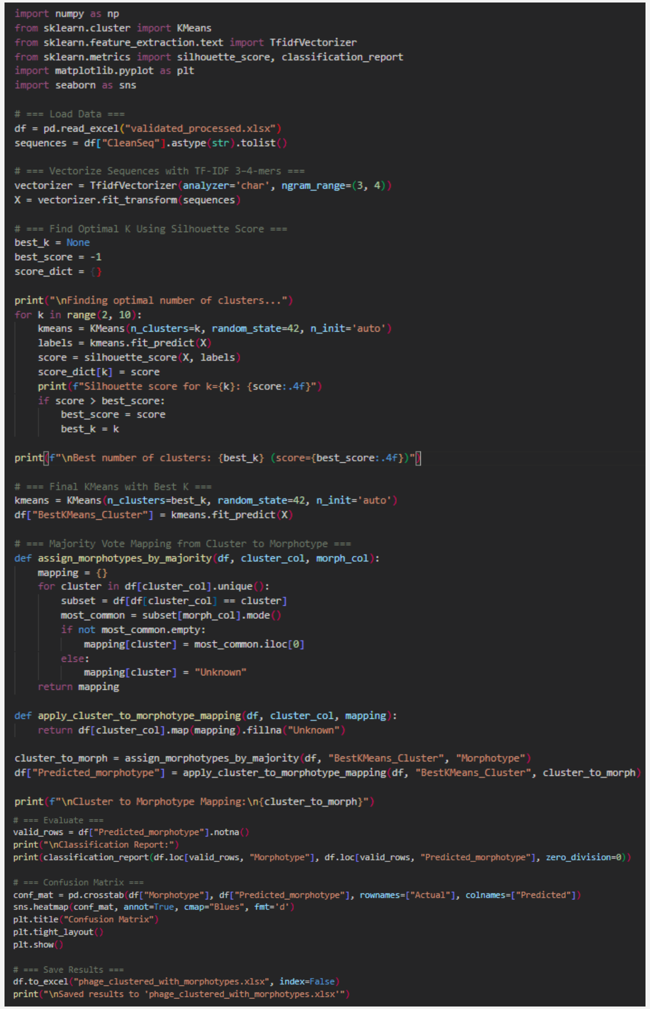
Workflow data processing - VALIDATED:
- Add FASTA format when missing
- Create a column with only aa to remove duplicates
- Create column with FASTA format to create FASTA file
- Create column with duplicated identifiers when found
- Remove duplicate lines based on sequence
- Create FASTA file and Excel file



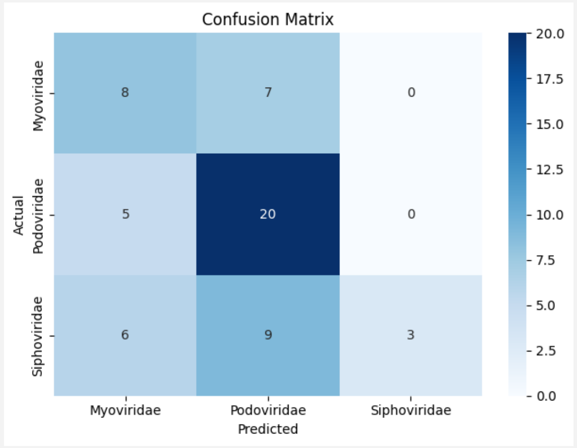
hage	Host	Protein Type	RBP Accession	Phage accession nr	Morphotype	CleanSeq	FAS
Bact 7C	Escherichia	tailspike	YP_004782195.1	NC_015933.1	Podoviridae	MNEMFSQGG	>YP_004782195.1 ge vB_Ec
Ba	Escherichia	tailspike	YP_009168860.1	NC_027994.1	Podoviridae	MSSGCGDVLS	>YP_00916 K1H]M
B	cherichia	tail spike	Q04830	NC_007636.1	Podoviridae	MSTITQFPSGI	>Q04 TITQF
	hia	tailspike	AEM91897.1	JN593240	Siphoviridae	MTRNVEELE	0]M
		tailspike	AEM91898.1	JN593240	Siphoviridae	MISO	]M
		spike	AEM91899.1	JN593240	Siphoviridae		
			YP_009880322.1	NC_049433.1	Siphovir		
			NP_112089.1	NC_002730.1	Si		
			YP_512279.1	NC_007804.2			
			99572.1	FR775895.2			
			.1	NC_008			
				N			
							]MLDI
Bacter							phage KP32]
Bacteriophage							phage EP335]MSR
Bacteriophage PhiE							ber [Bacteriophage PhiEc
Bacteriophage vB_EcoP_G							ber [Bacteriophage vB_EcoP_G
Bacteriophage vB_EcoS_AKFV33							2457.1 tail fiber [Bacteriophage vB_Ec
Bacteriophage K1-5	Es						654147.1 tailspike [Bacteriophage K1-5]MAK
Bacteriophage vB_CkoM_CkP1	Salmone						>QQG32074.1 tail fiber [Bacteriophage vB_CkoM_
Bacteriophage T4	Enter						P13332 tail fiber [Bacteriophage T4]MTLLSPGIEL
Bacteriophage vB_SauM_LM12							907.1 tail fiber [Bacteriophage vB_SauM_
Bacteriophage KpnM6E1							il fiber [Bacteriophage KpnM6E1]
Bacteriophage T5							Bacteriophage T5]MSFFAGKI
Bacteriophage M							acteriophage Mu]MF
Bacterioph							iophage P2]MT
Sal							age vB_
				NC_0			
			0002f0c	NC_002747			
			pdb_00005e7f	NC_002703			
			pdb_0000q0pdk6	NC_004166			
			pdb_00008xci	NC_001416	Sipho		
		er	pdb_00005efv	NC_004615	Siphoviridae		
		ail fiber	pdb_00007qlr	CP011970.1	Siphoviridae	M	ci
		putative structural	pdb_00004rga	NC_027120.1	Siphoviridae	GVLKGI	hag
	robacteria	long tail fiber	P03744	NC_000866.4	Myoviridae	MATLKQIQFK	> ATLK
Ba	Enterobacteria	short tail fiber	P10930	NC_000866.4	Myoviridae	MSNNTYQHV	>P1093 MSNN
Bac	Escherichia	tail fiber	P0DJY6	AF083977.1	Myoviridae	MPKSTIIQNLC	>P0DJY6 tail fi KSTIIQN

Validated RBPs after Processement with Python

Validated RBPs Analysis K-means and Confusion Matrix



```python
import numpy as np
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import silhouette_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# === Load Data ===
df = pd.read_excel("validated_processed.xlsx")
sequences = df["CleanSeq"].astype(str).tolist()

# === Vectorize Sequences with TF-IDF 3-4-mers ===
vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)

# === Find Optimal K Using Silhouette Score ===
best_k = None
best_score = -1
score_dict = {}

print("\nFinding optimal number of clusters...")
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    labels = kmeans.fit_predict(X)
    score = silhouette_score(X, labels)
    score_dict[k] = score
    print(f"Silhouette score for k={k}: {score:.4f}")
    if score > best_score:
        best_score = score
        best_k = k

print(f"\nBest number of clusters: {best_k} (score={best_score:.4f})")

# === Final KMeans with Best K ===
kmeans = KMeans(n_clusters=best_k, random_state=42, n_init='auto')
df["BestKMeans_Cluster"] = kmeans.fit_predict(X)

# === Majority Vote Mapping from Cluster to Morphotype ===
def assign_morphotypes_by_majority(df, cluster_col, morph_col):
    mapping = {}
    for cluster in df[cluster_col].unique():
        subset = df[df[cluster_col] == cluster]
        most_common = subset[morph_col].mode()
        if not most_common.empty:
            mapping[cluster] = most_common.iloc[0]
        else:
            mapping[cluster] = "Unknown"
    return mapping

def apply_cluster_to_morphotype_mapping(df, cluster_col, mapping):
    return df[cluster_col].map(mapping).fillna("Unknown")

cluster_to_morph = assign_morphotypes_by_majority(df, "BestKMeans_Cluster", "Morphotype")
df["Predicted_morphotype"] = apply_cluster_to_morphotype_mapping(df, "BestKMeans_Cluster", cluster_to_morph)

print(f"\nCluster to Morphotype Mapping:\n{cluster_to_morph}")
# === Evaluate ===
valid_rows = df["Predicted_morphotype"].notna()
print("\nClassification Report:")
print(classification_report(df.loc[valid_rows, "Morphotype"], df.loc[valid_rows, "Predicted_morphotype"], zero_division=0))

# === Confusion Matrix ===
conf_mat = pd.crosstab(df["Morphotype"], df["Predicted_morphotype"], rownames=["Actual"], colnames=["Predicted"])
sns.heatmap(conf_mat, annot=True, cmap="Blues", fmt='d')
plt.title("Confusion Matrix")
plt.tight_layout()
plt.show()

# === Save Results ===
df.to_excel("phage_clustered_with_morphotypes.xlsx", index=False)
print("\nSaved results to 'phage_clustered_with_morphotypes.xlsx'")
```



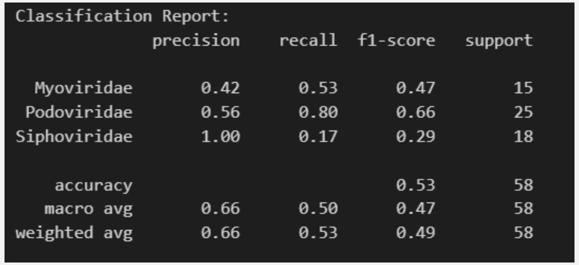
Performance Analysis:

Podoviridae:

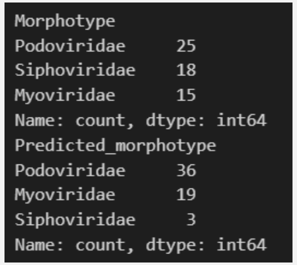
Highest recall and f1-score
↓
Reliably identified

Myoviridae and Siphoviridae:

Weaker performance;
Low recall for Siphoviridae
↓
Potential issues in classification

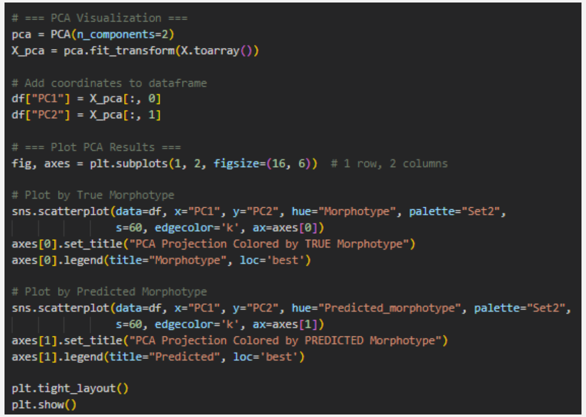


```
Classification Report:
              precision    recall  f1-score   support

  Myoviridae       0.42      0.53      0.47        15
 Podoviridae       0.56      0.80      0.66        25
Siphoviridae       1.00      0.17      0.29        18

    accuracy                           0.53        58
   macro avg       0.66      0.50      0.47        58
weighted avg       0.66      0.53      0.49        58
```



```
Morphotype
Podoviridae     25
Siphoviridae    18
Myoviridae      15
Name: count, dtype: int64
Predicted_morphotype
Podoviridae     36
Myoviridae      19
Siphoviridae     3
Name: count, dtype: int64
```

Validated RBPs – PCA and Clusters



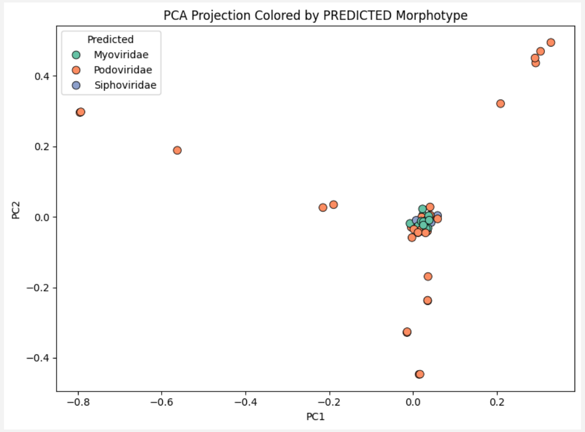
```python
# === PCA Visualization ===
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X.toarray())

# Add coordinates to dataframe
df["PC1"] = X_pca[:, 0]
df["PC2"] = X_pca[:, 1]

# === Plot PCA Results ===
fig, axes = plt.subplots(1, 2, figsize=(16, 6))  # 1 row, 2 columns

# Plot by True Morphotype
sns.scatterplot(data=df, x="PC1", y="PC2", hue="Morphotype", palette="Set2",
                s=60, edgecolor='k', ax=axes[0])
axes[0].set_title("PCA Projection Colored by TRUE Morphotype")
axes[0].legend(title="Morphotype", loc='best')

# Plot by Predicted Morphotype
sns.scatterplot(data=df, x="PC1", y="PC2", hue="Predicted_morphotype", palette="Set2",
                s=60, edgecolor='k', ax=axes[1])
axes[1].set_title("PCA Projection Colored by PREDICTED Morphotype")
axes[1].legend(title="Predicted", loc='best')

plt.tight_layout()
plt.show()
```

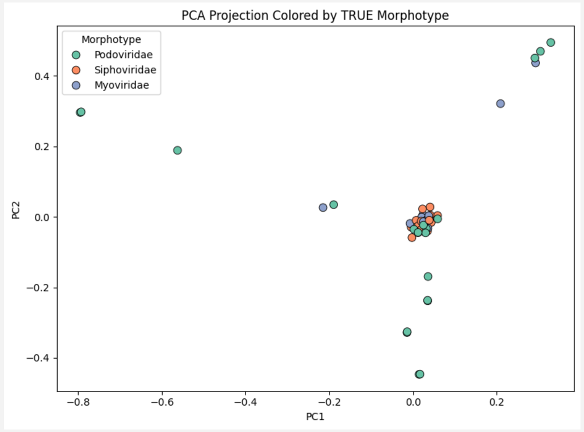
PCA - TRUE Morphotype Clusters:
Podoviridae: Well-separated
Siphoviridae: Cluster closer to Myoviridae
Myoviridae: Not clearly distinct; Slightly overlapping with Siphoviridae

- Distinct clustering for Podoviridae
- Confusion/blending between Siphoviridae and Myoviridae

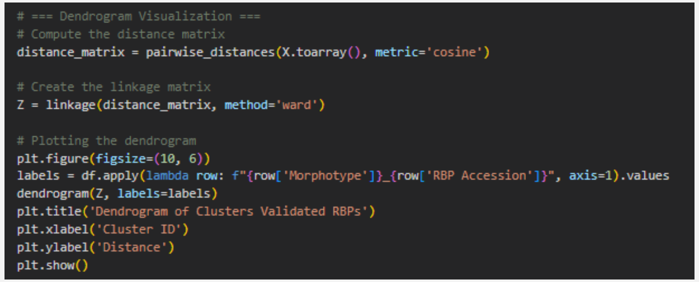
PCA - PREDICTED Morphotype Clusters:
Podoviridae: New clustering observed, likely misclassified from the TRUE representation.
Siphoviridae: Predicted model seems less accurate
Myoviridae: Some appear clustered together with Podoviridae

- Predictive model not capture morphotype distinctions effectively

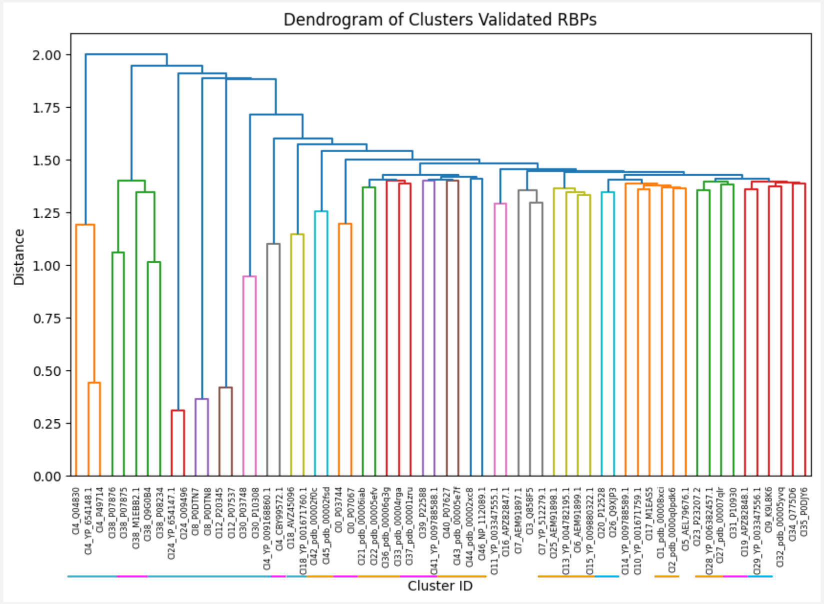




1st Dataset – Dendrogram K-means



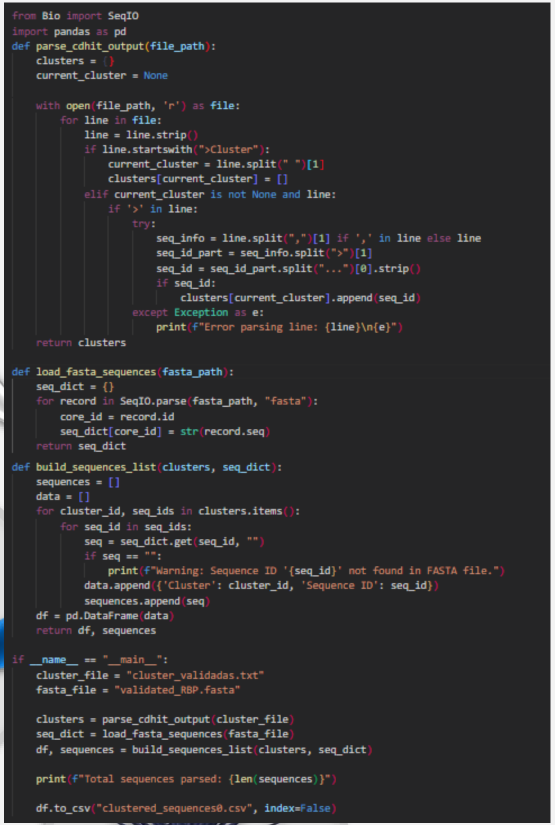
```
# === Dendrogram Visualization ===
# Compute the distance matrix
distance_matrix = pairwise_distances(X.toarray(), metric='cosine')

# Create the linkage matrix
Z = linkage(distance_matrix, method='ward')

# Plotting the dendrogram
plt.figure(figsize=(10, 6))
labels = df.apply(lambda row: f"{row['Morphotype']}_{row['RBP Accession']}", axis=1).values
dendrogram(Z, labels=labels)
plt.title('Dendrogram of Clusters Validated RBPs')
plt.xlabel('Cluster ID')
plt.ylabel('Distance')
plt.show()
```



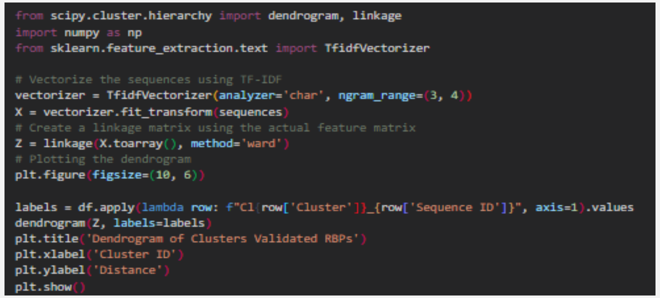
For global compositional similarity - identifying potential functional groups:

- Cluster shape: Compact, evenly sized clusters
- Separation between clusters: Distinct groups shared sequence features
- Reflecting functional or structural similarity, even among non-homologous proteins - spacing on the left side indicate relationships among RBPs
- Clustering suggests distinct groups among the validated RBPs – some by morphotype

1st Dataset – Validated RBPs CD-HIT



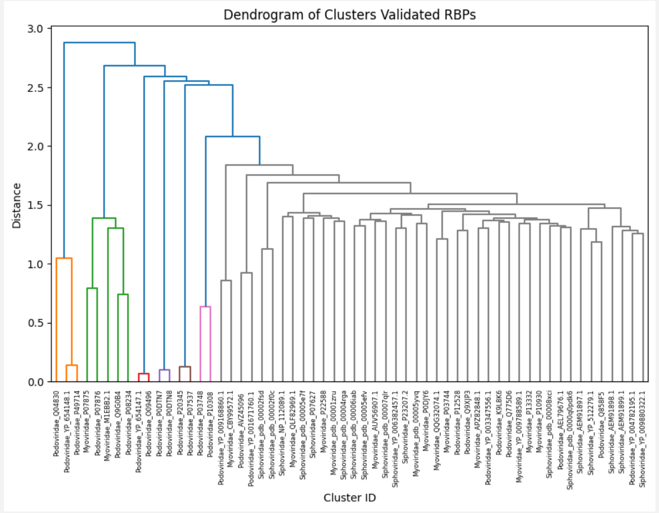
```python
from Bio import SeqIO
import pandas as pd
def parse_cdhit_output(file_path):
    clusters = {}
    current_cluster = None

    with open(file_path, 'r') as file:
        for line in file:
            line = line.strip()
            if line.startswith(">Cluster"):
                current_cluster = line.split(" ")[1]
                clusters[current_cluster] = []
            elif current_cluster is not None and line:
                if '>' in line:
                    try:
                        seq_info = line.split(",")[1] if ',' in line else line
                        seq_id_part = seq_info.split(">")[1]
                        seq_id = seq_id_part.split("...")[0].strip()
                        if seq_id:
                            clusters[current_cluster].append(seq_id)
                    except Exception as e:
                        print(f"Error parsing line: {line}\n{e}")
    return clusters

def load_fasta_sequences(fasta_path):
    seq_dict = {}
    for record in SeqIO.parse(fasta_path, "fasta"):
        core_id = record.id
        seq_dict[core_id] = str(record.seq)
    return seq_dict

def build_sequences_list(clusters, seq_dict):
    sequences = []
    data = []
    for cluster_id, seq_ids in clusters.items():
        for seq_id in seq_ids:
            seq = seq_dict.get(seq_id, "")
            if seq == "":
                print(f"Warning: Sequence ID '{seq_id}' not found in FASTA file.")
            data.append({'Cluster': cluster_id, 'Sequence ID': seq_id})
            sequences.append(seq)
    df = pd.DataFrame(data)
    return df, sequences

if __name__ == "__main__":
    cluster_file = "cluster_validadas.txt"
    fasta_file = "validated_RBP.fasta"

    clusters = parse_cdhit_output(cluster_file)
    seq_dict = load_fasta_sequences(fasta_file)
    df, sequences = build_sequences_list(clusters, seq_dict)

    print(f"Total sequences parsed: {len(sequences)}")

    df.to_csv("clustered_sequences0.csv", index=False)
```



```python
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorize the sequences using TF-IDF
vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)
# Create a linkage matrix using the actual feature matrix
Z = linkage(X.toarray(), method='ward')
# Plotting the dendrogram
plt.figure(figsize=(10, 6))

labels = df.apply(lambda row: f"Cl{row['Cluster']}_{row['Sequence ID']}", axis=1).values
dendrogram(Z, labels=labels)
plt.title('Dendrogram of Clusters Validated RBPs')
plt.xlabel('Cluster ID')
plt.ylabel('Distance')
plt.show()
```

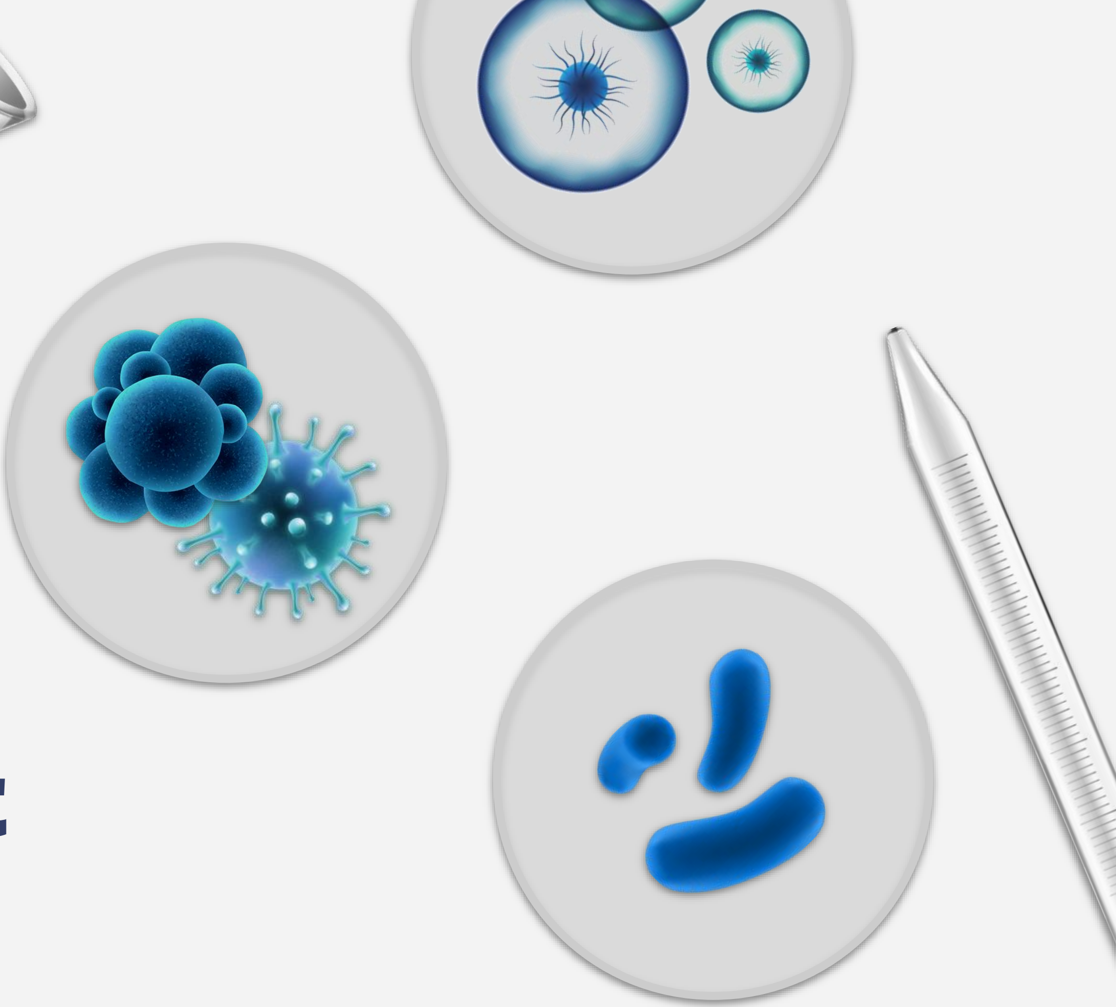
For redundancy removal and homology grouping:

- Pairwise sequence identity
- Cluster shape: Broader clusters
- Separation: Some branches merging at higher distances = low sequence similarity
- Bifurcations at grey represent a bigger diversification in less time comparing with the pink cluster with same ancestry
- Clustering suggests distinct groups among the validated RBPs – some by morphotype



4.2

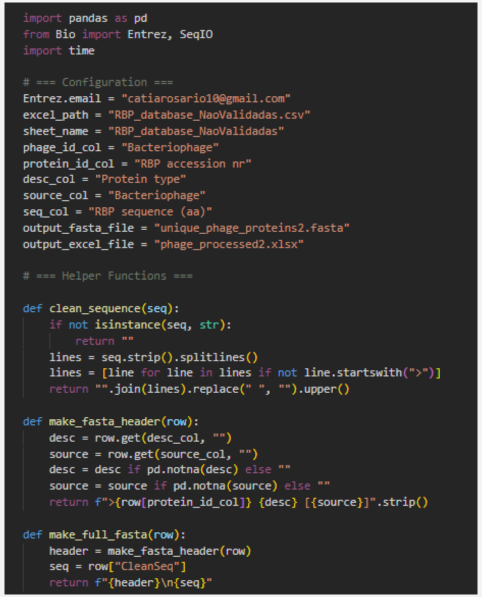
2nd Dataset



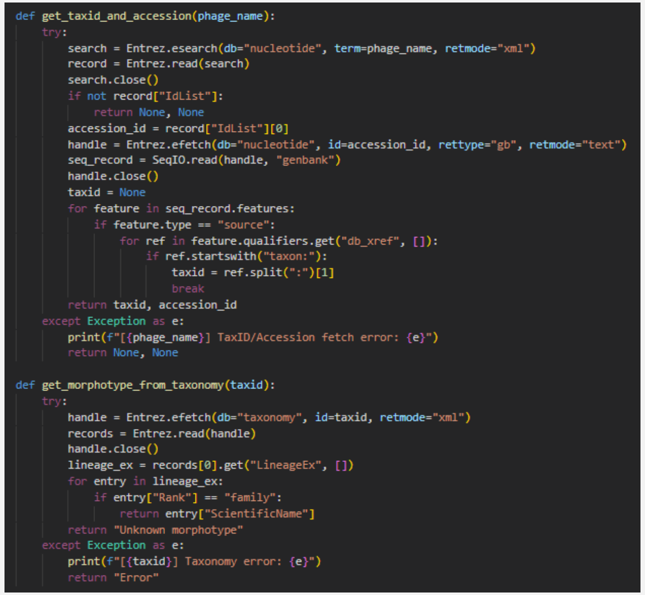
Phage accession nr		RBP accession nr	Protein type	RBP sequence (aa)	RBP Sequence (nr)	RBP Delineation	Host	seq 200	CleanSeq		catePhageIDs
1718840		0P0IE19	Tail fiber protein	MDKLEEFSLNGPKNTDGL	ATGGATAAGTTAGAAG	202	Acinetoba	MDKLEEFSI	MDKLEEFSLNG		acter phage Ab105-1phi]MD
1718840		63	Tail fiber protein	MATNWNAVLNNTNNFN	ATGGCAACAAACTGGA	684	Acinetoba	MATNWNA	MATNW		cter phage Ab105-1phi]M
1718842			tail fibers protein	MANLVFKFSWDHRPFPYI	ATGGCAAATCTTGTTTTT	451	Acinetoba	MANLVFKF	M		r phage Ab105-3phi]MA
2500826			l fibers protein	MILEGVYPSFLKGVSQQTF	ATGATTCTTGAGGGAGT	763	Acinetoba	MILEG			phage AbKT21phiIII]M
2015804			tail fiber prot	MANPIFIPIAFAANGIKNP	GTGGCTAATCCTATATTT	276	Acinetoba				tobacter phage AbP2
2015804			in	MTNPTLITTPFAENGDKN	ATGACCAATCCAACACT	776	Aci				hage AbP2]MTNPT
2282400				MLNEYEDKVVEVIVDVPK	ATGCTTAACGAATACGA	9					e ABPH49]MLNE
2282400				RVNVNPEFYAAVD	GTGGCAGATCCAAGAGT						ABPH49]MADP
228240				WAQQDIEL	ATGGCACGTCCATCGGT						phage ABPH4
25078				KKM	ATGACCGCCTTCTGGCG						TJ]MTAFWR
25078					TGGATAAGTTAGA						]MDKLEEF
1913571					TGAT						ge AM24]MADVLV
1913571											er phage AM24]MARINK
1913571	Ac										tobacter phage AM24]MEQLVLI
1187128	Acinetob										tein [Acinetobacter phage AP22]MAN
1221835	Acinetobacter p										r [Acinetobacter phage Bphi-B1251]MLPFAS
1647373	Acinetobacter phage										tail spike protein [Acinetobacter phage Fri1]MNILF
2500136	Acinetobacter phage Henu										.1 long distal tail fiber subunit [Acinetobacter phage Her
2500136	Acinetobacter phage Henu6										003892.1 long tail fiber proximal subunit [Acinetobacter phage
2500136	Acinetobacter phage Henu6	QA							KF		>QAU04025.1 short tail fibers protein [Acinetobacter phage Henu6]M
1735582	Acinetobacter phage IME200	A0A0P0IJ7							FTETVV		>A0A0P0IJ73 Tail fiber protein [Acinetobacter phage IME200]MNILRS
1243183	Acinetobacter phage IME-AB2	AFV51556.1							NPIFIPIAFA		>AFV51556.1 tail fiber protein-like protein [Acinetobacter phage IM
1243183	Acinetobacter phage IME-AB2	K4P093							TTPFA		>K4P093 Putative tail fiber [Acinetobacter phage IME-AB2]MTNPTLIT
1243183	Acinetobacter phage IME-AB2	K4									K4P0C2 Putative tail-fiber/lysozyme protein [Acinetobacter phage
2301662	Acinetobacter phage KARL-1										658.1 long distal tail fiber subunit [Acinetobacter phage KAR
2301662	Acinetobacter phage KA										long tail fiber proximal subunit [Acinetobacter phage K
2301662	Acinetobacter pha										il fibers protein [Acinetobacter phage KARL-1]M
1792222	Acinetobact										tein [Acinetobacter phage LZ35]MANPIFI
1792222	Acine										Acinetobacter phage LZ35]MTNPTLI
1406779											ter phage Petty]MAYVEKEMN
691318											ge phiAB1]MNILRSFTE
1698439					ATATAC						]MNILRSFTETV
1406					ATGGCTCGTATTGATT						RIDWQPLQ
22021				LDPNGV	ATGTCAAAGACACTTAA						9]MSKTLKIEV
1481186				DFENIWASEGC	ATGGTAGCTATAGTAAA						er phage vB_A
1481186				KVLTGNETVSIVGNFA	ATGTCAAAAGTATTAAC						cter phage vB_A
2315466			prot	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2211					acter phage vB_A
2315467			iber prot	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2211	Acinet				bacter phage vB_A
2315458			er protein	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2223	Acinetoba	M			phage vB_AbaM_B09
2163899			tail spike protein	MGAMKIPTGGFGDYSITV	ATGGGAGCAATGAAGAT	863	Acinetoba	MGAMK			er phage vB_AbaM_B9]
2163899			Tail fiber	MAEQKINELPITPPSNFTD	ATGGCAGAACAAAAAAT	452	Acinetoba	MAEQKINE	MAE		e vB_AbaM_B9]MAEQKI
2163899		GTT1	Tail fiber	MSVVSRPDYSVGVWASN	ATGTCTGTAGTTAGCAG	348	Acinetoba	MSVVSRPD	MSVVSRP		ge vB_AbaM_B9]MSVVSR
1837876		0A172Q097	Tail fiber protein	MELKCSLQDGRGLYHVTS	ATGGAACTGAAATGTTC	574	Acinetoba	MELKCSLQ	MELKCSLQDGR	>	bacter phage vB_AbaM_ME

2nd Dataset – Computationally Predicted RBPs
]

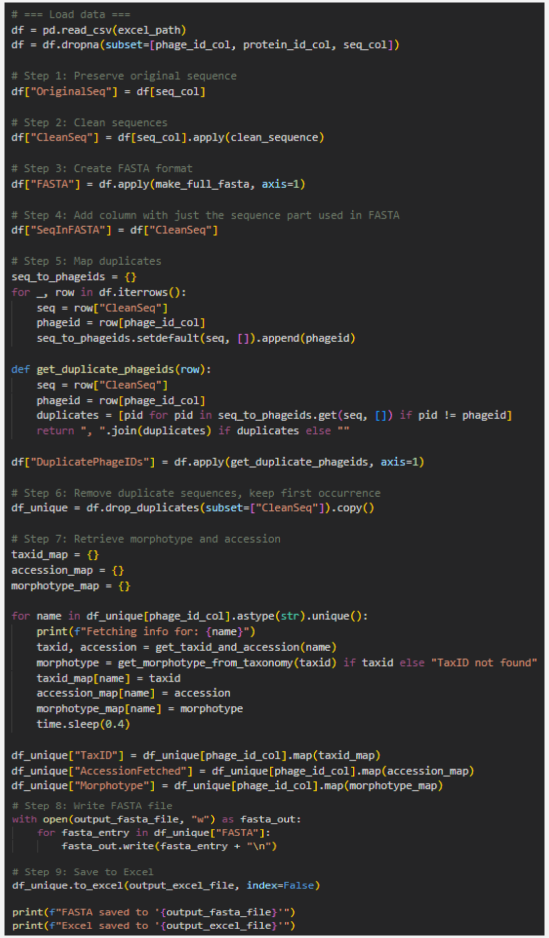
2nd Dataset Processement with Python



```python
import pandas as pd
from Bio import Entrez, SeqIO
import time

# === Configuration ===
Entrez.email = "catiarosario10@gmail.com"
excel_path = "RBP_database_NaoValidadas.csv"
sheet_name = "RBP_database_NaoValidadas"
phage_id_col = "Bacteriophage"
protein_id_col = "RBP accession nr"
desc_col = "Protein type"
source_col = "Bacteriophage"
seq_col = "RBP sequence (aa)"
output_fasta_file = "unique_phage_proteins2.fasta"
output_excel_file = "phage_processed2.xlsx"

# === Helper Functions ===

def clean_sequence(seq):
    if not isinstance(seq, str):
        return ""
    lines = seq.strip().splitlines()
    lines = [line for line in lines if not line.startswith(">")]
    return "".join(lines).replace(" ", "").upper()

def make_fasta_header(row):
    desc = row.get(desc_col, "")
    source = row.get(source_col, "")
    desc = desc if pd.notna(desc) else ""
    source = source if pd.notna(source) else ""
    return f">{row[protein_id_col]} {desc} [{source}]".strip()

def make_full_fasta(row):
    header = make_fasta_header(row)
    seq = row["CleanSeq"]
    return f"{header}\n{seq}"
```



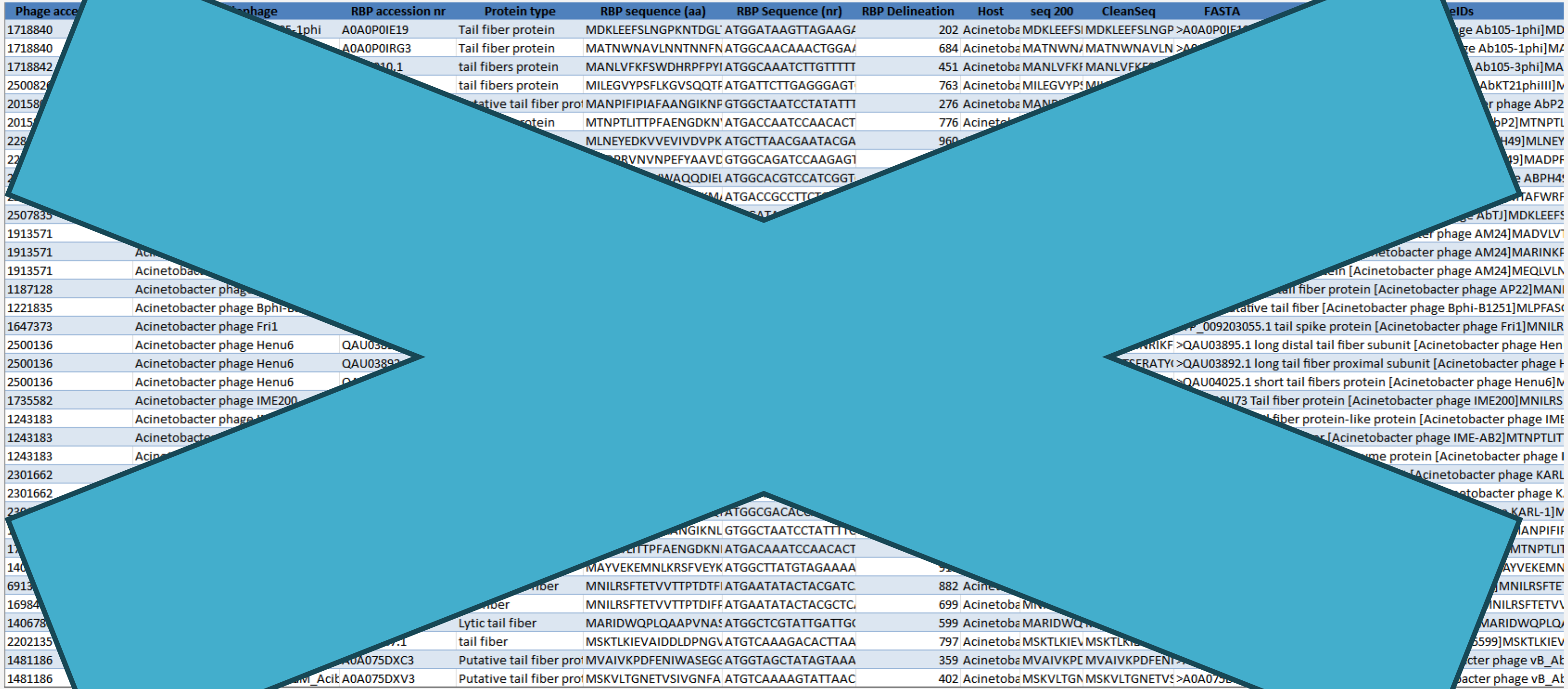
```python
def get_taxid_and_accession(phage_name):
    try:
        search = Entrez.esearch(db="nucleotide", term=phage_name, retmode="xml")
        record = Entrez.read(search)
        search.close()
        if not record["IdList"]:
            return None, None
        accession_id = record["IdList"][0]
        handle = Entrez.efetch(db="nucleotide", id=accession_id, rettype="gb", retmode="text")
        seq_record = SeqIO.read(handle, "genbank")
        handle.close()
        taxid = None
        for feature in seq_record.features:
            if feature.type == "source":
                for ref in feature.qualifiers.get("db_xref", []):
                    if ref.startswith("taxon:"):
                        taxid = ref.split(":")[1]
                        break
        return taxid, accession_id
    except Exception as e:
        print(f"[{phage_name}] TaxID/Accession fetch error: {e}")
        return None, None

def get_morphotype_from_taxonomy(taxid):
    try:
        handle = Entrez.efetch(db="taxonomy", id=taxid, retmode="xml")
        records = Entrez.read(handle)
        handle.close()
        lineage_ex = records[0].get("LineageEx", [])
        for entry in lineage_ex:
            if entry["Rank"] == "family":
                return entry["ScientificName"]
        return "Unknown morphotype"
    except Exception as e:
        print(f"[{taxid}] Taxonomy error: {e}")
        return "Error"
```



```python
# === Load data ===
df = pd.read_csv(excel_path)
df = df.dropna(subset=[phage_id_col, protein_id_col, seq_col])

# Step 1: Preserve original sequence
df["OriginalSeq"] = df[seq_col]

# Step 2: Clean sequences
df["CleanSeq"] = df[seq_col].apply(clean_sequence)

# Step 3: Create FASTA format
df["FASTA"] = df.apply(make_full_fasta, axis=1)

# Step 4: Add column with just the sequence part used in FASTA
df["SeqInFASTA"] = df["CleanSeq"]

# Step 5: Map duplicates
seq_to_phageids = {}
for _, row in df.iterrows():
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    seq_to_phageids.setdefault(seq, []).append(phageid)

def get_duplicate_phageids(row):
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    duplicates = [pid for pid in seq_to_phageids.get(seq, []) if pid != phageid]
    return ", ".join(duplicates) if duplicates else ""

df["DuplicatePhageIDs"] = df.apply(get_duplicate_phageids, axis=1)

# Step 6: Remove duplicate sequences, keep first occurrence
df_unique = df.drop_duplicates(subset=["CleanSeq"]).copy()

# Step 7: Retrieve morphotype and accession
taxid_map = {}
accession_map = {}
morphotype_map = {}

for name in df_unique[phage_id_col].astype(str).unique():
    print(f"Fetching info for: {name}")
    taxid, accession = get_taxid_and_accession(name)
    morphotype = get_morphotype_from_taxonomy(taxid) if taxid else "TaxID not found"
    taxid_map[name] = taxid
    accession_map[name] = accession
    morphotype_map[name] = morphotype
    time.sleep(0.4)

df_unique["TaxID"] = df_unique[phage_id_col].map(taxid_map)
df_unique["AccessionFetched"] = df_unique[phage_id_col].map(accession_map)
df_unique["Morphotype"] = df_unique[phage_id_col].map(morphotype_map)
# Step 8: Write FASTA file
with open(output_fasta_file, "w") as fasta_out:
    for fasta_entry in df_unique["FASTA"]:
        fasta_out.write(fasta_entry + "\n")

# Step 9: Save to Excel
df_unique.to_excel(output_excel_file, index=False)

print(f"FASTA saved to '{output_fasta_file}'")
print(f"Excel saved to '{output_excel_file}'")
```

Workflow data processing - NON-VALIDATED:

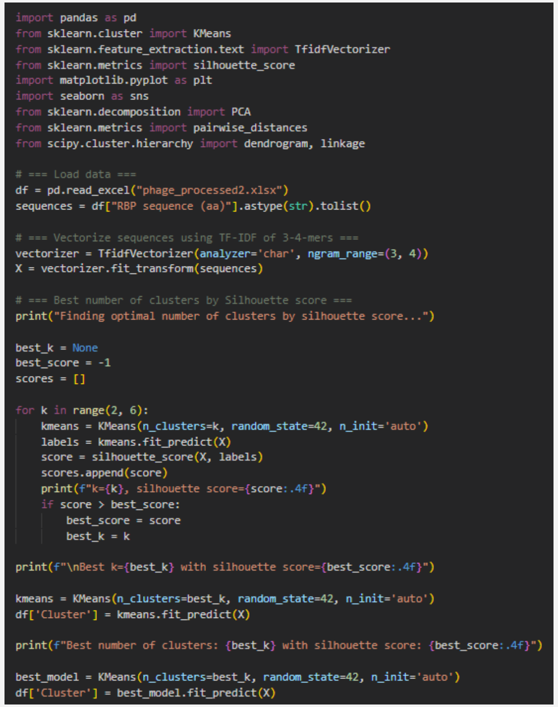
- Fetch the id of bacteriophages from name
- Add column with ids
- Add FASTA format if missing
- Create column with only aa to remove duplicates
- Create column with FASTA format
- Create new column with duplicated identifiers
- Remove duplicate lines based on sequence
- Retrieve morphotypes
- Create FASTA file and Excel file



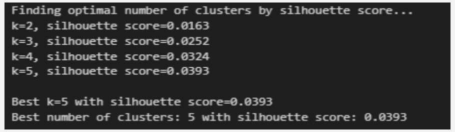
Phage acce	phage	RBP accession nr	Protein type	RBP sequence (aa)	RBP Sequence (nr)	RBP Delineation	Host	seq 200	CleanSeq	FASTA	eIDs
1718840	5-1phi	A0A0P0IE19	Tail fiber protein	MDKLEEFSLNGPKNTDGL	ATGGATAAGTTAGAAGA	202	Acinetoba	MDKLEEFSI	MDKLEEFSLNGP	>A0A0P0IE1	ge Ab105-1phi]MD
1718840		A0A0P0IRG3	Tail fiber protein	MATNWNAVLNNTNNFN	ATGGCAACAAACTGGAA	684	Acinetoba	MATNWNA	MATNWNAVLN	>A0	ge Ab105-1phi]MA
1718842		010.1	tail fibers protein	MANLVFKFSWDHRPFPYI	ATGGCAAATCTTGTTTTT	451	Acinetoba	MANLVFKF	MANLVFKF		Ab105-3phi]MA
2500826			tail fibers protein	MILEGVYPSFLKGVSQTF	ATGATTCTTGAGGGAGT	763	Acinetoba	MILEGVYPS	MI		AbKT21phiIII]M
20158			tative tail fiber prot	MANPIFIPIAFAANGIKNF	GTGGCTAATCCTATATTT	276	Acinetoba	MAN			r phage AbP2
2015			otein	MTNPTLITTPFAENGDKN	ATGACCAATCCAACACT	776	Acinet				bP2]MTNPTL
228				MLNEYEDKVVEVIVDVPK	ATGCTTAACGAATACGA	960					H49]MLNEY
22				PRVNVNPEFYAAVD	GTGGCAGATCCAAGAGT						49]MADPF
2				WAQQDIEL	ATGGCACGTCCATCGGT						e ABPH4
				M	ATGACCGCCTTCT						TAFWRF
2507835					AT						e AbTJ]MDKLEEFS
1913571											r phage AM24]MADVLV
1913571	Ac									etobacter phage AM24]MARINKF	
1913571	Acinetobac									ein [Acinetobacter phage AM24]MEQLVLN	
1187128	Acinetobacter phag									ail fiber protein [Acinetobacter phage AP22]MANI	
1221835	Acinetobacter phage Bphi-B									tative tail fiber [Acinetobacter phage Bphi-B1251]MLPFASC	
1647373	Acinetobacter phage Fri1									P_009203055.1 tail spike protein [Acinetobacter phage Fri1]MNILR	
2500136	Acinetobacter phage Henu6	QAU03							NRIKF	>QAU03895.1 long distal tail fiber subunit [Acinetobacter phage Hen	
2500136	Acinetobacter phage Henu6	QAU03892							TSERATY	>QAU03892.1 long tail fiber proximal subunit [Acinetobacter phage H	
2500136	Acinetobacter phage Henu6	QA								>QAU04025.1 short tail fibers protein [Acinetobacter phage Henu6]M	
1735582	Acinetobacter phage IME200									0U73 Tail fiber protein [Acinetobacter phage IME200]MNILRS	
1243183	Acinetobacter phage I									fiber protein-like protein [Acinetobacter phage IME	
1243183	Acinetobacte									r [Acinetobacter phage IME-AB2]MTNPTLIT	
1243183	Acin									me protein [Acinetobacter phage I	
2301662										[Acinetobacter phage KARL	
2301662										etobacter phage K	
230					ATGGCGACAC						KARL-1]M
				ANGIKNL	GTGGCTAATCCTATTTTC						ANPIFIF
17				LITTPFAENGDKNI	ATGACAAATCCAACACT						MTNPTLIT
140				MAYVEKEMNLKRSFVEYK	ATGGCTTATGTAGAAAA	91					AYVEKEMN
6913			iber	MNILRSFTETVVTTPTDTFI	ATGAATATACTACGATC	882	Acine				]MNILRSFTE
16984			iber	MNILRSFTETVVTTPTDIFF	ATGAATATACTACGCTC	699	Acinetoba	MN			NILRSFTETV
140678			Lytic tail fiber	MARIDWQPLQAAPVNAS	ATGGCTCGTATTGATTGG	599	Acinetoba	MARIDWQ			MARIDWQPLQ
2202135		7.1	tail fiber	MSKTLKIEVAIDDLDPNGV	ATGTCAAAGACACTTAA	797	Acinetoba	MSKTLKIEV	MSKTLKI		5599]MSKTLKIEV
1481186		A0A075DXC3	Putative tail fiber prot	MVAIVKPDFENIWASEGC	ATGGTAGCTATAGTAAA	359	Acinetoba	MVAIVKPD	MVAIVKPDFENI	>	cter phage vB_Ab
1481186	M_Acib	A0A075DXV3	Putative tail fiber prot	MSKVLTGNETVSIVGNFA	ATGTCAAAAGTATTAAC	402	Acinetoba	MSKVLTGN	MSKVLTGNETVS	>A0A075	acter phage vB_Ab

Computationally Predicted RBPs after Processement with Python

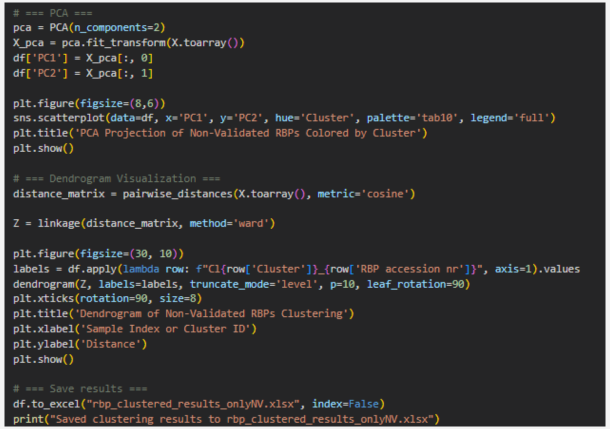
2nd Dataset Analysis – K-means and PCA



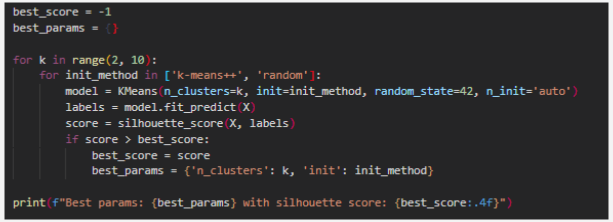
```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.metrics import pairwise_distances
from scipy.cluster.hierarchy import dendrogram, linkage

# === Load data ===
df = pd.read_excel("phage_processed2.xlsx")
sequences = df["RBP sequence (aa)"].astype(str).tolist()

# === Vectorize sequences using TF-IDF of 3-4-mers ===
vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)

# === Best number of clusters by Silhouette score ===
print("Finding optimal number of clusters by silhouette score...")

best_k = None
best_score = -1
scores = []

for k in range(2, 6):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    labels = kmeans.fit_predict(X)
    score = silhouette_score(X, labels)
    scores.append(score)
    print(f"k={k}, silhouette score={score:.4f}")
    if score > best_score:
        best_score = score
        best_k = k

print(f"\nBest k={best_k} with silhouette score={best_score:.4f}")

kmeans = KMeans(n_clusters=best_k, random_state=42, n_init='auto')
df['Cluster'] = kmeans.fit_predict(X)

print(f"Best number of clusters: {best_k} with silhouette score: {best_score:.4f}")

best_model = KMeans(n_clusters=best_k, random_state=42, n_init='auto')
df['Cluster'] = best_model.fit_predict(X)
```



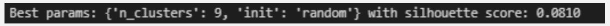
```
Finding optimal number of clusters by silhouette score...
k=2, silhouette score=0.0163
k=3, silhouette score=0.0252
k=4, silhouette score=0.0324
k=5, silhouette score=0.0393

Best k=5 with silhouette score=0.0393
Best number of clusters: 5 with silhouette score: 0.0393
```



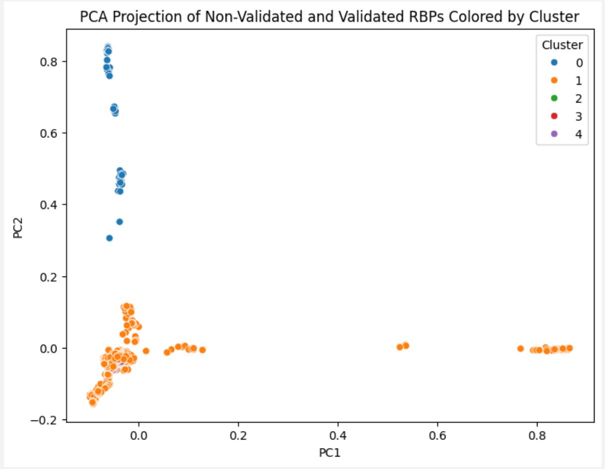
```python
# === PCA ===
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X.toarray())
df['PC1'] = X_pca[:, 0]
df['PC2'] = X_pca[:, 1]

plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='PC1', y='PC2', hue='Cluster', palette='tab10', legend='full')
plt.title('PCA Projection of Non-Validated RBPs Colored by Cluster')
plt.show()

# === Dendrogram Visualization ===
distance_matrix = pairwise_distances(X.toarray(), metric='cosine')

Z = linkage(distance_matrix, method='ward')

plt.figure(figsize=(30, 10))
labels = df.apply(lambda row: f"Cl{row['Cluster']}_{row['RBP accession nr']}", axis=1).values
dendrogram(Z, labels=labels, truncate_mode='level', p=10, leaf_rotation=90)
plt.xticks(rotation=90, size=8)
plt.title('Dendrogram of Non-Validated RBPs Clustering')
plt.xlabel('Sample Index or Cluster ID')
plt.ylabel('Distance')
plt.show()

# === Save results ===
df.to_excel("rbp_clustered_results_onlyNV.xlsx", index=False)
print("Saved clustering results to rbp_clustered_results_onlyNV.xlsx")
```



```python
best_score = -1
best_params = {}

for k in range(2, 10):
    for init_method in ['k-means++', 'random']:
        model = KMeans(n_clusters=k, init=init_method, random_state=42, n_init='auto')
        labels = model.fit_predict(X)
        score = silhouette_score(X, labels)
        if score > best_score:
            best_score = score
            best_params = {'n_clusters': k, 'init': init_method}

print(f"Best params: {best_params} with silhouette score: {best_score:.4f}")
```



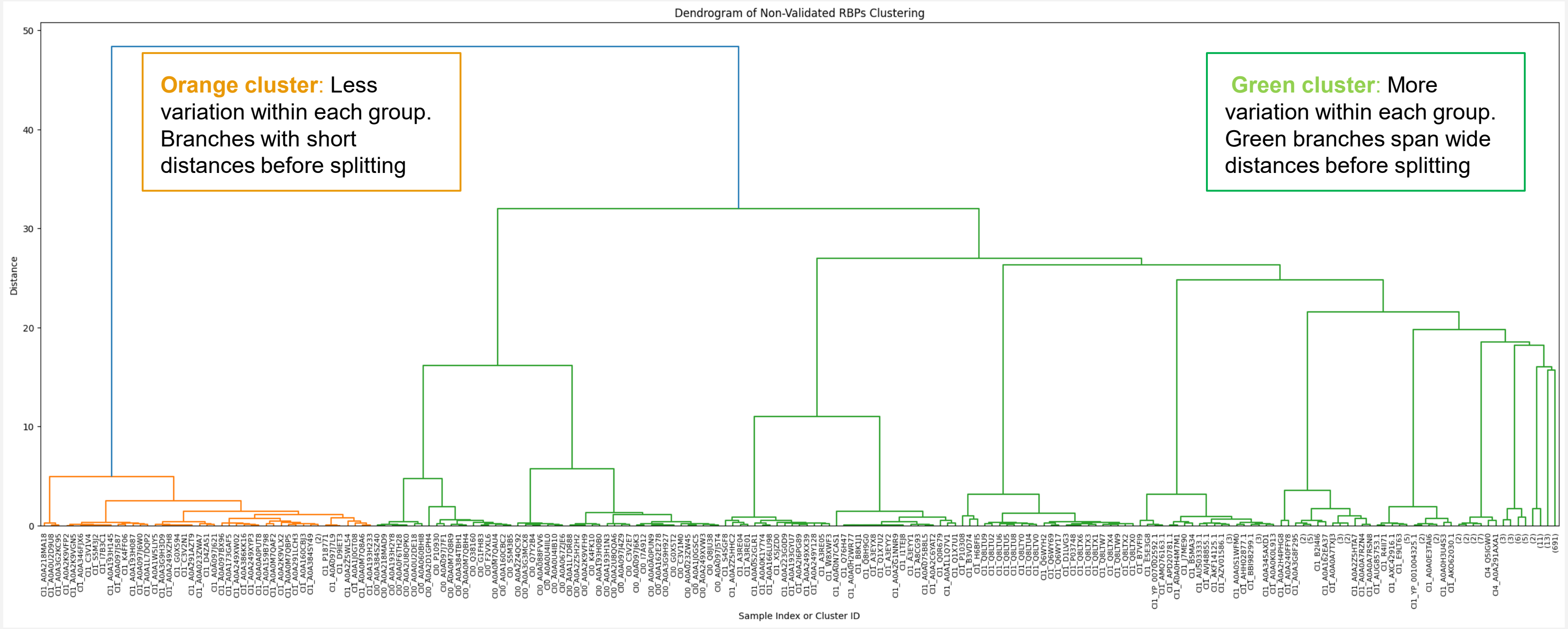
```
Best params: {'n_clusters': 9, 'init': 'random'} with silhouette score: 0.0810
```



Cluster 0:
Separated from others → distinct characteristics
Cluster 1:
Larger group with some spread along both axes
Cluster 2, Cluster 3, Cluster 4:
Less populated and overlap → similarity in characteristics

Interpretation:
- Distinct clusters → Non-Validated RBPs may have different characteristics
- **Cluster 0** is notably distinct, potentially representing a unique group of RBPs

2nd Dataset Analysis – Dendrogram



Orange cluster: Less variation within each group. Branches with short distances before splitting

Green cluster: More variation within each group. Green branches span wide distances before splitting

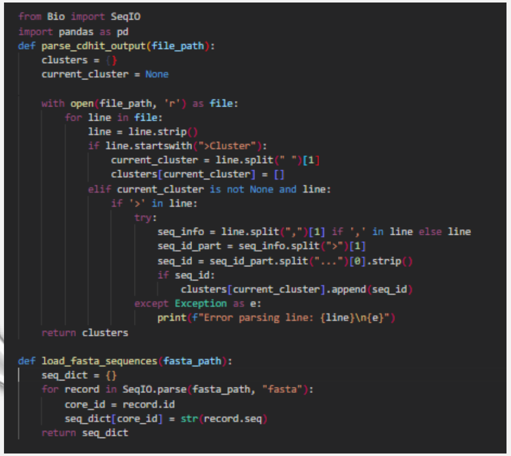
- Cluster shape: Broad
- Separation between clusters: High
- Cohesion between sub-groups
- Nº Clusters: Low

- Dissimilar sequences that share functional elements

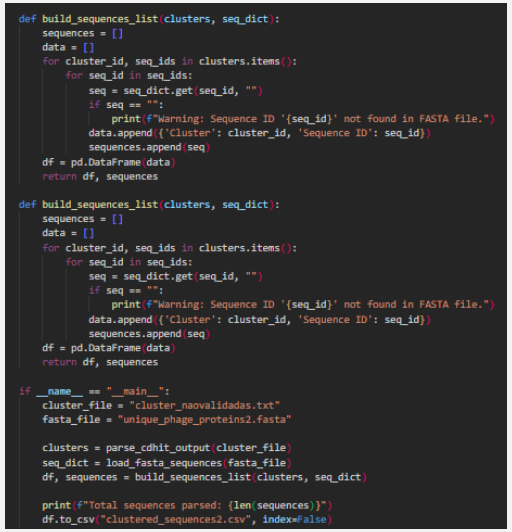
These clusters likely represent:

- Possibly novel or divergent RBPs
- Groupings of functionally analogous

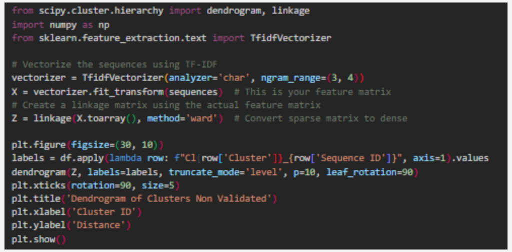
2nd Dataset – Non-Validated RBPs CD-HIT



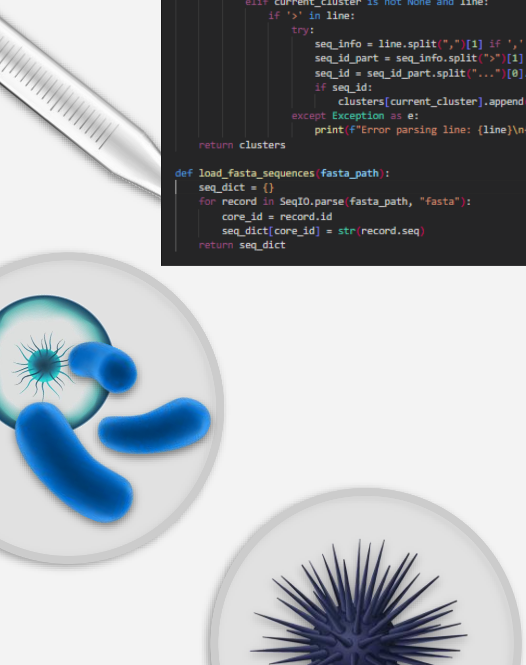
```python
from Bio import SeqIO
import pandas as pd
def parse_cdhit_output(file_path):
    clusters = {}
    current_cluster = None

    with open(file_path, 'r') as file:
        for line in file:
            line = line.strip()
            if line.startswith(">Cluster"):
                current_cluster = line.split(" ")[1]
                clusters[current_cluster] = []
            elif current_cluster is not None and line:
                if '>' in line:
                    try:
                        seq_info = line.split(",")[1] if ',' in line else line
                        seq_id_part = seq_info.split(">")[1]
                        seq_id = seq_id_part.split("...")[0].strip()
                        if seq_id:
                            clusters[current_cluster].append(seq_id)
                    except Exception as e:
                        print(f"Error parsing line: {line}\n{e}")
    return clusters

def load_fasta_sequences(fasta_path):
    seq_dict = {}
    for record in SeqIO.parse(fasta_path, "fasta"):
        core_id = record.id
        seq_dict[core_id] = str(record.seq)
    return seq_dict
```



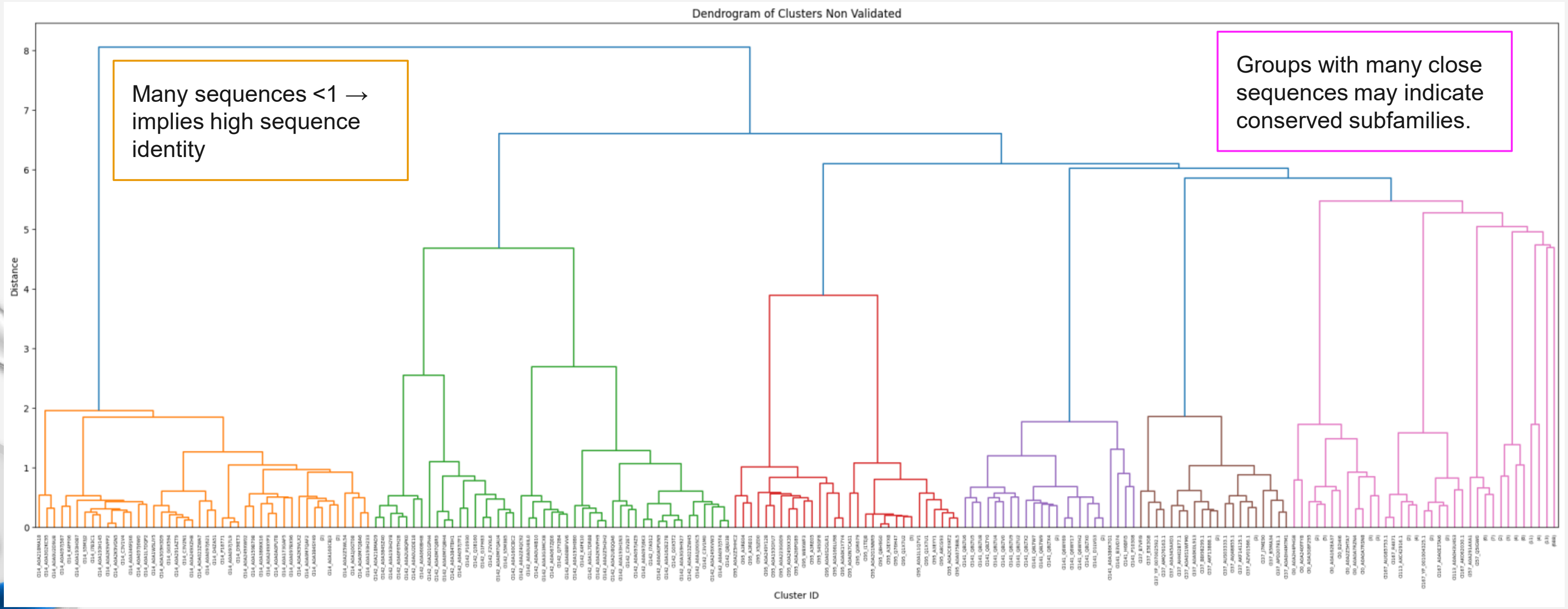
```python
def build_sequences_list(clusters, seq_dict):
    sequences = []
    data = []
    for cluster_id, seq_ids in clusters.items():
        for seq_id in seq_ids:
            seq = seq_dict.get(seq_id, "")
            if seq == "":
                print(f"Warning: Sequence ID '{seq_id}' not found in FASTA file.")
            data.append({'Cluster': cluster_id, 'Sequence ID': seq_id})
            sequences.append(seq)
    df = pd.DataFrame(data)
    return df, sequences

def build_sequences_list(clusters, seq_dict):
    sequences = []
    data = []
    for cluster_id, seq_ids in clusters.items():
        for seq_id in seq_ids:
            seq = seq_dict.get(seq_id, "")
            if seq == "":
                print(f"Warning: Sequence ID '{seq_id}' not found in FASTA file.")
            data.append({'Cluster': cluster_id, 'Sequence ID': seq_id})
            sequences.append(seq)
    df = pd.DataFrame(data)
    return df, sequences

if __name__ == "__main__":
    cluster_file = "cluster_naovalidadas.txt"
    fasta_file = "unique_phage_proteins2.fasta"

    clusters = parse_cdhit_output(cluster_file)
    seq_dict = load_fasta_sequences(fasta_file)
    df, sequences = build_sequences_list(clusters, seq_dict)

    print(f"Total sequences parsed: {len(sequences)}")
    df.to_csv("clustered_sequences2.csv", index=False)
```

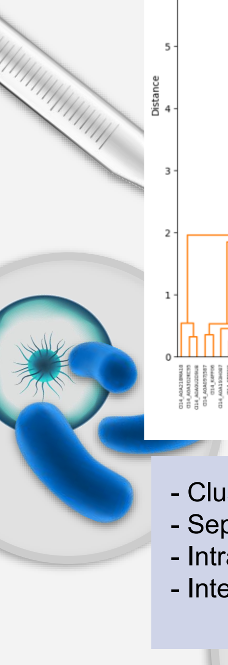


```python
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorize the sequences using TF-IDF
vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)  # This is your feature matrix
# Create a linkage matrix using the actual feature matrix
Z = linkage(X.toarray(), method='ward')  # Convert sparse matrix to dense

plt.figure(figsize=(30, 10))
labels = df.apply(lambda row: f"Cl{row['Cluster']}_{row['Sequence ID']}", axis=1).values
dendrogram(Z, labels=labels, truncate_mode='level', p=10, leaf_rotation=90)
plt.xticks(rotation=90, size=5)
plt.title('Dendrogram of Clusters Non Validated')
plt.xlabel('Cluster ID')
plt.ylabel('Distance')
plt.show()
```



2nd Dataset – Non-Validated RBPs CD-HIT





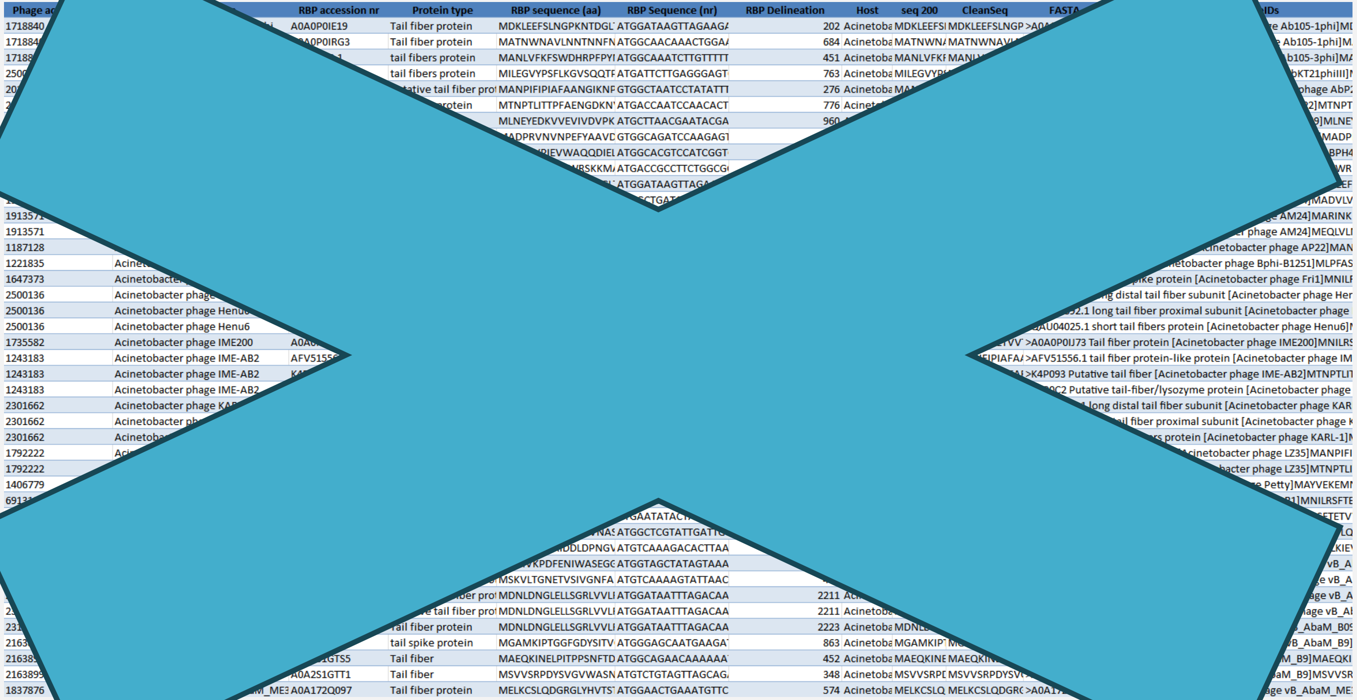
- Cluster shape: Compact, narrow (short branches within)
- Separation between clusters: High (clear distinction)
- Intra-cluster similarity: High (identity ≥ 40%)
- Inter-cluster difference: High (hard cutoff)

Each cluster could correspond to:

- One phage RBP subtype,
- A known structural domain or tail fiber class,
- Evolutionary lineages with strong sequence conservation.

4.3

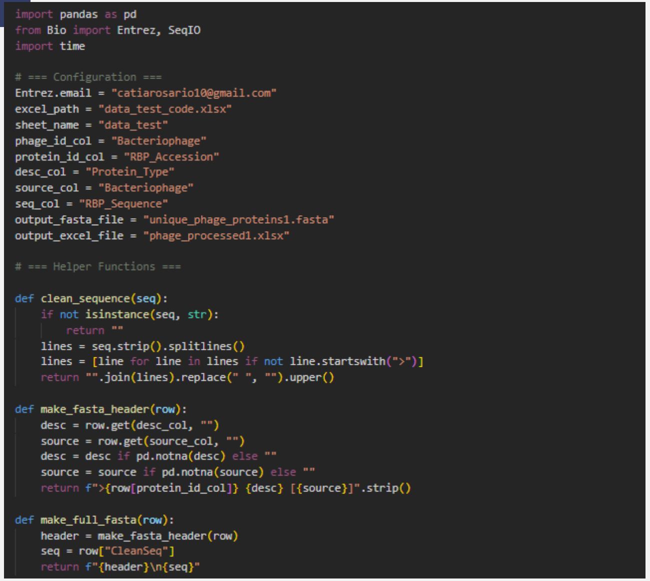
3rd Dataset



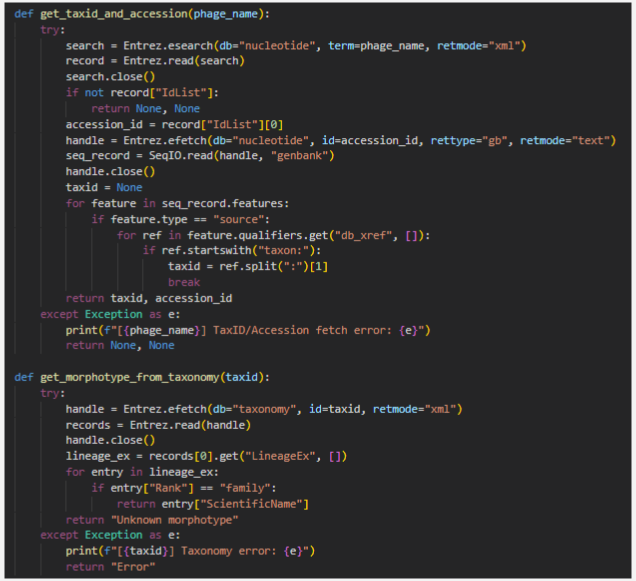
Phage ac	[illegible]	RBP accession nr	Protein type	RBP sequence (aa)	RBP Sequence (nr)	RBP Delineation	Host	seq 200	CleanSeq	FASTA	IDs
1718840	[illegible]	A0A0P0IE19	Tail fiber protein	MDKLEEFSLNGPKNTDGL	ATGGATAAGTTAGAAGA	202	Acinetoba	MDKLEEFS	MDKLEEFSLNGP	>A0A	e Ab105-1phi]MD
171884	[illegible]	A0P0IRG3	Tail fiber protein	MATNWNAVLNNTNNFN	ATGGCAACAAACTGGAA	684	Acinetoba	MATNWNA	MATNWNAV	[illegible]	e Ab105-1phi]MA
17188	[illegible]	[illegible]	tail fibers protein	MANLVFKFSWDHRPFPYI	ATGGCAAATCTTGTTTTT	451	Acinetoba	MANLVFKF	MANL	[illegible]	Ab105-3phi]MA
2500	[illegible]	[illegible]	tail fibers protein	MILEGVYPSFLKGVSQQTF	ATGATTCTTGAGGGAGT	763	Acinetoba	MILEGVYP	[illegible]	[illegible]	bKT21phiIII]M
20	[illegible]	[illegible]	tative tail fiber prot	MANPIFIPIAFAANGIKNP	GTGGCTAATCCTATATTT	276	Acinetoba	MA	[illegible]	[illegible]	phage AbP2
2	[illegible]	[illegible]	protein	MTNPTLITTPFAENGDKN	ATGACCAATCCAACACT	776	Acinet	[illegible]	[illegible]	[illegible]	2]MTNPT
[illegible]	[illegible]	[illegible]	[illegible]	MLNEYEDKVVEVIVDVPK	ATGCTTAACGAATACGA	960	[illegible]	[illegible]	[illegible]	[illegible]	9]MLNEY
[illegible]	[illegible]	[illegible]	[illegible]	ADPRVNVNPEFYAAVD	GTGGCAGATCCAAGAGT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	MADP
[illegible]	[illegible]	[illegible]	[illegible]	IEVWAQQDIEL	ATGGCACGTCCATCGGT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	BPH4
[illegible]	[illegible]	[illegible]	[illegible]	RSKKMA	ATGACCGCCTTCTGGCG	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	WR
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ATGGATAAGTTAGA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	EF
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	CTGAT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	]MADVLV
191357	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ge AM24]MARINK
1913571	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	r phage AM24]MEQLVLI
1187128	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	cinetobacter phage AP22]MAN
1221835	Acinet	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	etobacter phage Bphi-B1251]MLPFAS
1647373	Acinetobacte	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	pike protein [Acinetobacter phage Fri1]MNILF
2500136	Acinetobacter phage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ng distal tail fiber subunit [Acinetobacter phage Her
2500136	Acinetobacter phage Henu	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	92.1 long tail fiber proximal subunit [Acinetobacter phage
2500136	Acinetobacter phage Henu6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	AU04025.1 short tail fibers protein [Acinetobacter phage Henu6]M
1735582	Acinetobacter phage IME200	A0A0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ETVV	>A0A0P0IJ73 Tail fiber protein [Acinetobacter phage IME200]MNILRS	
1243183	Acinetobacter phage IME-AB2	AFV5155	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	FIPIAFAA	>AFV51556.1 tail fiber protein-like protein [Acinetobacter phage IM	
1243183	Acinetobacter phage IME-AB2	K4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	A	>K4P093 Putative tail fiber [Acinetobacter phage IME-AB2]MTNPTLIT	
1243183	Acinetobacter phage IME-AB2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0C2 Putative tail-fiber/lysozyme protein [Acinetobacter phage	
2301662	Acinetobacter phage KA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1 long distal tail fiber subunit [Acinetobacter phage KAR	
2301662	Acinetobacter ph	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	il fiber proximal subunit [Acinetobacter phage K	
2301662	Acinetoba	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	rs protein [Acinetobacter phage KARL-1]M
1792222	Aci	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	cinetobacter phage LZ35]MANPIFI
1792222	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	bacter phage LZ35]MTNPTLI
1406779	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	e Petty]MAYVEKEMN
6913	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1]MNILRSFTE
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	GAATATAC	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	SFETETV
[illegible]	[illegible]	[illegible]	[illegible]	NAS	ATGGCTCGTATTGATT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	LQ
[illegible]	[illegible]	[illegible]	[illegible]	IDDLDPNGV	ATGTCAAAGACACTTAA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	LKIEV
[illegible]	[illegible]	[illegible]	[illegible]	VKPDFENIWASEGG	ATGGTAGCTATAGTAAA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	vB_A
[illegible]	[illegible]	[illegible]	[illegible]	MSKVLTGNETVSIVGNFA	ATGTCAAAAGTATTAAC	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ge vB_A
[illegible]	[illegible]	[illegible]	ber prot	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2211	Aci	[illegible]	[illegible]	[illegible]	age vB_A
23	[illegible]	[illegible]	e tail fiber prot	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2211	Acinetoba	[illegible]	[illegible]	[illegible]	age vB_Ab
231	[illegible]	[illegible]	ail fiber protein	MDNLDNGLELLSGRLVVLI	ATGGATAATTTAGACAA	2223	Acinetoba	MDNL	[illegible]	[illegible]	B_AbaM_B09
2163	[illegible]	[illegible]	tail spike protein	MGAMKIPTGGFGDYSITV	ATGGGAGCAATGAAGAT	863	Acinetoba	MGAMKIPT	MG	[illegible]	vB_AbaM_B9]
216389	[illegible]	1GTS5	Tail fiber	MAEQKINELPITPPSNFTD	ATGGCAGAACAAAAAAT	452	Acinetoba	MAEQKINE	MAEQKIN	[illegible]	M_B9]MAEQKI
2163899	[illegible]	A0A2S1GTT1	Tail fiber	MSVVSRPDYSVGVWASN	ATGTCTGTAGTTAGCAGA	348	Acinetoba	MSVVSRPD	MSVVSRPDYSV	[illegible]	aM_B9]MSVVSR
1837876	M_ME3	A0A172Q097	Tail fiber protein	MELKCSLQDGRGLYHVTST	ATGGAACTGAAATGTTC	574	Acinetoba	MELKCSLQ	MELKCSLQDGRG	>A0A17	age vB_AbaM_ME

3nd Dataset – Computationally Predicted and Validated RBPs

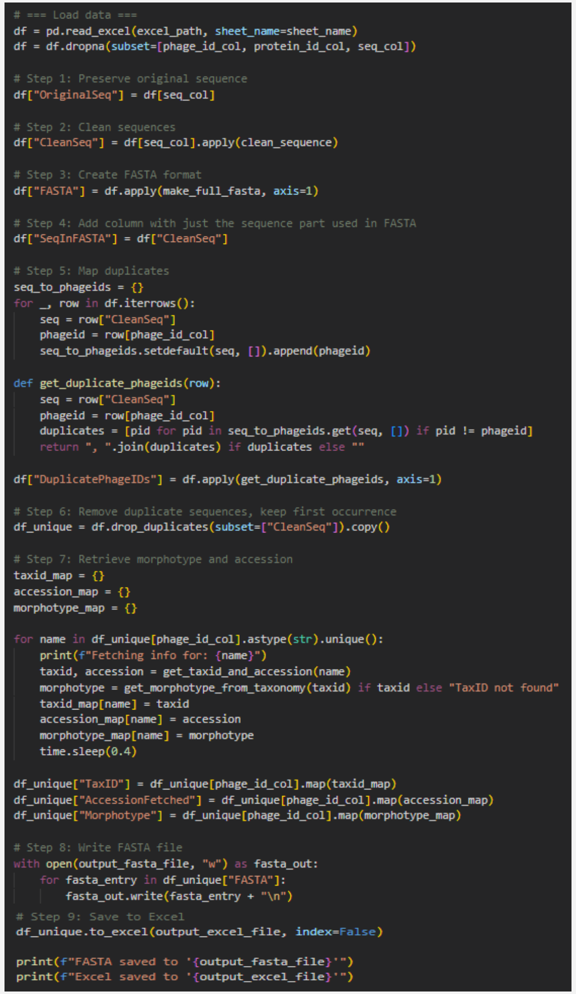
3rd Dataset Processement with Python



```python
import pandas as pd
from Bio import Entrez, SeqIO
import time

# === Configuration ===
Entrez.email = "catiarosario10@gmail.com"
excel_path = "data_test_code.xlsx"
sheet_name = "data_test"
phage_id_col = "Bacteriophage"
protein_id_col = "RBP_Accession"
desc_col = "Protein_Type"
source_col = "Bacteriophage"
seq_col = "RBP_Sequence"
output_fasta_file = "unique_phage_proteins1.fasta"
output_excel_file = "phage_processed1.xlsx"

# === Helper Functions ===

def clean_sequence(seq):
    if not isinstance(seq, str):
        return ""
    lines = seq.strip().splitlines()
    lines = [line for line in lines if not line.startswith(">")]
    return "".join(lines).replace(" ", "").upper()

def make_fasta_header(row):
    desc = row.get(desc_col, "")
    source = row.get(source_col, "")
    desc = desc if pd.notna(desc) else ""
    source = source if pd.notna(source) else ""
    return f">{row[protein_id_col]} {desc} [{source}]".strip()

def make_full_fasta(row):
    header = make_fasta_header(row)
    seq = row["CleanSeq"]
    return f"{header}\n{seq}"
```



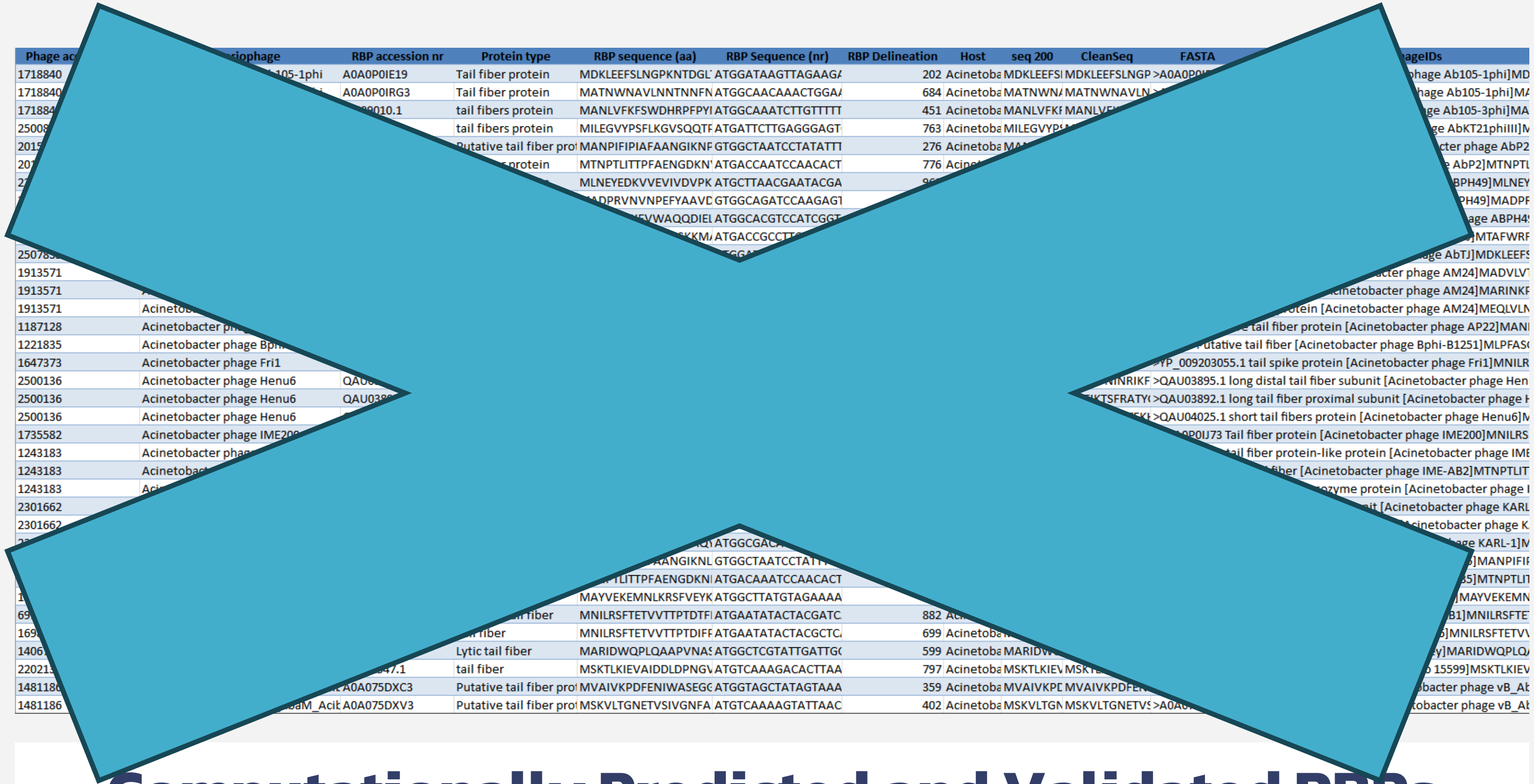
```python
def get_taxid_and_accession(phage_name):
    try:
        search = Entrez.esearch(db="nucleotide", term=phage_name, retmode="xml")
        record = Entrez.read(search)
        search.close()
        if not record["IdList"]:
            return None, None
        accession_id = record["IdList"][0]
        handle = Entrez.efetch(db="nucleotide", id=accession_id, rettype="gb", retmode="text")
        seq_record = SeqIO.read(handle, "genbank")
        handle.close()
        taxid = None
        for feature in seq_record.features:
            if feature.type == "source":
                for ref in feature.qualifiers.get("db_xref", []):
                    if ref.startswith("taxon:"):
                        taxid = ref.split(":")[1]
                        break
        return taxid, accession_id
    except Exception as e:
        print(f"[{phage_name}] TaxID/Accession fetch error: {e}")
        return None, None

def get_morphotype_from_taxonomy(taxid):
    try:
        handle = Entrez.efetch(db="taxonomy", id=taxid, retmode="xml")
        records = Entrez.read(handle)
        handle.close()
        lineage_ex = records[0].get("LineageEx", [])
        for entry in lineage_ex:
            if entry["Rank"] == "family":
                return entry["ScientificName"]
        return "Unknown morphotype"
    except Exception as e:
        print(f"[{taxid}] Taxonomy error: {e}")
        return "Error"
```



```python
# === Load data ===
df = pd.read_excel(excel_path, sheet_name=sheet_name)
df = df.dropna(subset=[phage_id_col, protein_id_col, seq_col])

# Step 1: Preserve original sequence
df["OriginalSeq"] = df[seq_col]

# Step 2: Clean sequences
df["CleanSeq"] = df[seq_col].apply(clean_sequence)

# Step 3: Create FASTA format
df["FASTA"] = df.apply(make_full_fasta, axis=1)

# Step 4: Add column with just the sequence part used in FASTA
df["SeqInFASTA"] = df["CleanSeq"]

# Step 5: Map duplicates
seq_to_phageids = {}
for _, row in df.iterrows():
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    seq_to_phageids.setdefault(seq, []).append(phageid)

def get_duplicate_phageids(row):
    seq = row["CleanSeq"]
    phageid = row[phage_id_col]
    duplicates = [pid for pid in seq_to_phageids.get(seq, []) if pid != phageid]
    return ", ".join(duplicates) if duplicates else ""

df["DuplicatePhageIDs"] = df.apply(get_duplicate_phageids, axis=1)

# Step 6: Remove duplicate sequences, keep first occurrence
df_unique = df.drop_duplicates(subset=["CleanSeq"]).copy()

# Step 7: Retrieve morphotype and accession
taxid_map = {}
accession_map = {}
morphotype_map = {}

for name in df_unique[phage_id_col].astype(str).unique():
    print(f"Fetching info for: {name}")
    taxid, accession = get_taxid_and_accession(name)
    morphotype = get_morphotype_from_taxonomy(taxid) if taxid else "TaxID not found"
    taxid_map[name] = taxid
    accession_map[name] = accession
    morphotype_map[name] = morphotype
    time.sleep(0.4)

df_unique["TaxID"] = df_unique[phage_id_col].map(taxid_map)
df_unique["AccessionFetched"] = df_unique[phage_id_col].map(accession_map)
df_unique["Morphotype"] = df_unique[phage_id_col].map(morphotype_map)

# Step 8: Write FASTA file
with open(output_fasta_file, "w") as fasta_out:
    for fasta_entry in df_unique["FASTA"]:
        fasta_out.write(fasta_entry + "\n")
# Step 9: Save to Excel
df_unique.to_excel(output_excel_file, index=False)

print(f"FASTA saved to '{output_fasta_file}'")
print(f"Excel saved to '{output_excel_file}'")
```

Workflow data processing – NON-VALIDATED:

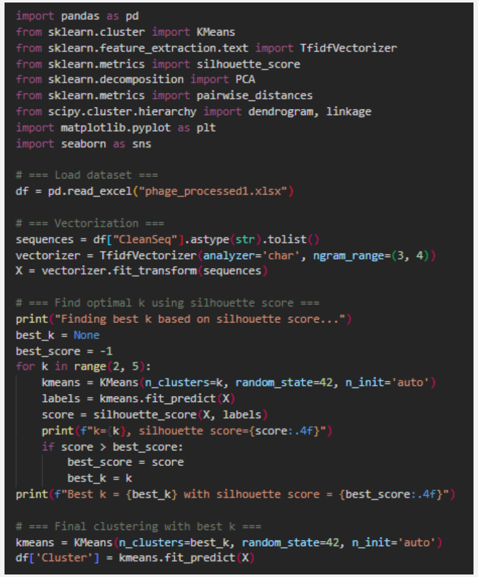
- Fetch the id of bacteriophages from name
- Add column with ids
- Add FASTA format if missing
- Create column with only aa to remove duplicates
- Create column with FASTA format
- Create new column with duplicated identifiers
- Remove duplicate lines based on sequence
- Retrieve morphotypes
- Create FASTA file and Excel file



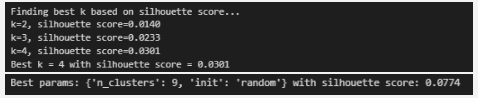
Phage ac	riophage	RBP accession nr	Protein type	RBP sequence (aa)	RBP Sequence (nr)	RBP Delineation	Host	seq 200	CleanSeq	FASTA	ageIDs
1718840	105-1phi	A0A0P0IE19	Tail fiber protein	MDKLEEFSLNGPKNTDGL	ATGGATAAGTTAGAAGA	202	Acinetoba	MDKLEEFSI	MDKLEEFSLNGP	>A0A0P0I	phage Ab105-1phi]MD
1718840		A0A0P0IRG3	Tail fiber protein	MATNWNAVLNNTNNFN	ATGGCAACAAACTGGAA	684	Acinetoba	MATNWNA	MATNWNAVLN	>	hage Ab105-1phi]MA
171884		09010.1	tail fibers protein	MANLVFKFSWDHRPFPYI	ATGGCAAATCTTGTTTTT	451	Acinetoba	MANLVFKF	MANLV		ge Ab105-3phi]MA
25008			tail fibers protein	MILEGVYPSFLKGVSQTF	ATGATTCTTGAGGGAGT	763	Acinetoba	MILEGVYPS			ge AbKT21phiIII]M
2015			Putative tail fiber prot	MANPIFIPIAFAANGIKNF	GTGGCTAATCCTATATTT	276	Acinetoba	MA			cter phage AbP2
201			protein	MTNPTLITTPFAENGDKN	ATGACCAATCCAACACT	776	Acin				AbP2]MTNPTL
2				MLNEYEDKVVEVIVDVPK	ATGCTTAACGAATACGA	96					BPH49]MLNEY
				ADPRVNVNPEFYAAVD	GTGGCAGATCCAAGAGT						PH49]MADPR
				EVWAQQDIEL	ATGGCACGTCCATCGGT						age ABPH4
				SKKMA	ATGACCGCCTT						]MTAFWRF
25078					TGGA						age AbTJ]MDKLEEFS
1913571											cter phage AM24]MADVLV
1913571	A										cinetobacter phage AM24]MARINKF
1913571	Acinetob										otein [Acinetobacter phage AM24]MEQLVLN
1187128	Acinetobacter ph										tail fiber protein [Acinetobacter phage AP22]MANI
1221835	Acinetobacter phage Bph										utative tail fiber [Acinetobacter phage Bphi-B1251]MLPFASC
1647373	Acinetobacter phage Fri1										YP_009203055.1 tail spike protein [Acinetobacter phage Fri1]MNILR
2500136	Acinetobacter phage Henu6	QAU0							NINRIKF	>QAU03895.1 long distal tail fiber subunit [Acinetobacter phage Hen	
2500136	Acinetobacter phage Henu6	QAU038							KTSFRATY(	>QAU03892.1 long tail fiber proximal subunit [Acinetobacter phage H	
2500136	Acinetobacter phage Henu6								EKI	>QAU04025.1 short tail fibers protein [Acinetobacter phage Henu6]M	
1735582	Acinetobacter phage IME200									0P0IJ73 Tail fiber protein [Acinetobacter phage IME200]MNILRS	
1243183	Acinetobacter phag										tail fiber protein-like protein [Acinetobacter phage IME
1243183	Acinetobac										fiber [Acinetobacter phage IME-AB2]MTNPTLIT
1243183	Aci										ozyme protein [Acinetobacter phage I
2301662											it [Acinetobacter phage KARL
2301662											cinetobacter phage K
2				RQ	ATGGCGAC						age KARL-1]M
				AANGIKNL	GTGGCTAATCCTAT						]MANPIFIF
				TLITTPFAENGDKNI	ATGACAAATCCAACACT						5]MTNPTLIT
1				MAYVEKEMNLKRSFVEYK	ATGGCTTATGTAGAAAA						]MAYVEKEMN
69			fiber	MNILRSFTETVVTTPTDTFI	ATGAATATACTACGATC	882	Ac				B1]MNILRSFTE
169			fiber	MNILRSFTETVVTTPTDIFF	ATGAATATACTACGCTCA	699	Acinetoba				5]MNILRSFTETV
1406			Lytic tail fiber	MARIDWQPLQAAPVNAS	ATGGCTCGTATTGATTGC	599	Acinetoba	MARIDW			y]MARIDWQPLQA
22021		47.1	tail fiber	MSKTLKIEVAIDDLDPNGV	ATGTCAAAGACACTTAA	797	Acinetoba	MSKTLKIEV	MSK		o 15599]MSKTLKIEV
148118		A0A075DXC3	Putative tail fiber prot	MVAIVKPDFENIWASEGC	ATGGTAGCTATAGTAAA	359	Acinetoba	MVAIVKPD	MVAIVKPDFE		bacter phage vB_Ab
1481186	aM_Acib	A0A075DXV3	Putative tail fiber prot	MSKVLTGNETVSIVGNFA	ATGTCAAAAGTATTAAC	402	Acinetoba	MSKVLTGN	MSKVLTGNETVS	>A0A0	tobacter phage vB_Ab

Computationally Predicted and Validated RBPs after Processement with Python

3rd Dataset Analysis – K-means and PCA

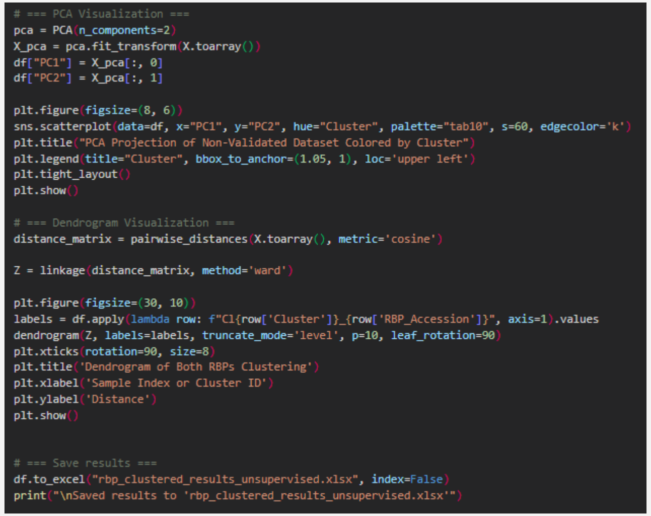


```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.metrics import pairwise_distances
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
import seaborn as sns

# === Load dataset ===
df = pd.read_excel("phage_processed1.xlsx")

# === Vectorization ===
sequences = df["CleanSeq"].astype(str).tolist()
vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)

# === Find optimal k using silhouette score ===
print("Finding best k based on silhouette score...")
best_k = None
best_score = -1
for k in range(2, 5):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    labels = kmeans.fit_predict(X)
    score = silhouette_score(X, labels)
    print(f"k={k}, silhouette score={score:.4f}")
    if score > best_score:
        best_score = score
        best_k = k
print(f"Best k = {best_k} with silhouette score = {best_score:.4f}")

# === Final clustering with best k ===
kmeans = KMeans(n_clusters=best_k, random_state=42, n_init='auto')
df['Cluster'] = kmeans.fit_predict(X)
```

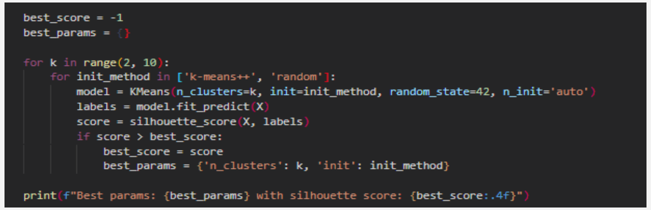


```
Finding best k based on silhouette score...
k=2, silhouette score=0.0140
k=3, silhouette score=0.0233
k=4, silhouette score=0.0301
Best k = 4 with silhouette score = 0.0301
```

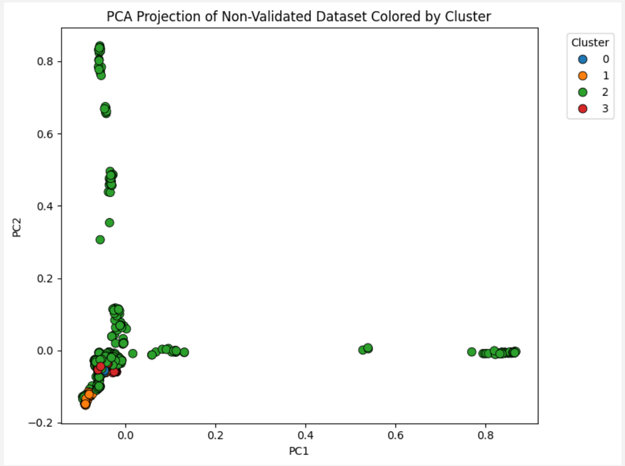
```
Best params: {'n_clusters': 9, 'init': 'random'} with silhouette score: 0.0774
```



```python
# === PCA Visualization ===
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X.toarray())
df["PC1"] = X_pca[:, 0]
df["PC2"] = X_pca[:, 1]

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x="PC1", y="PC2", hue="Cluster", palette="tab10", s=60, edgecolor='k')
plt.title("PCA Projection of Non-Validated Dataset Colored by Cluster")
plt.legend(title="Cluster", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# === Dendrogram Visualization ===
distance_matrix = pairwise_distances(X.toarray(), metric='cosine')

Z = linkage(distance_matrix, method='ward')

plt.figure(figsize=(30, 10))
labels = df.apply(lambda row: f"Cl{row['Cluster']}_{row['RBP_Accession']}", axis=1).values
dendrogram(Z, labels=labels, truncate_mode='level', p=10, leaf_rotation=90)
plt.xticks(rotation=90, size=8)
plt.title('Dendrogram of Both RBPs Clustering')
plt.xlabel('Sample Index or Cluster ID')
plt.ylabel('Distance')
plt.show()


# === Save results ===
df.to_excel("rbp_clustered_results_unsupervised.xlsx", index=False)
print("\nSaved results to 'rbp_clustered_results_unsupervised.xlsx'")
```



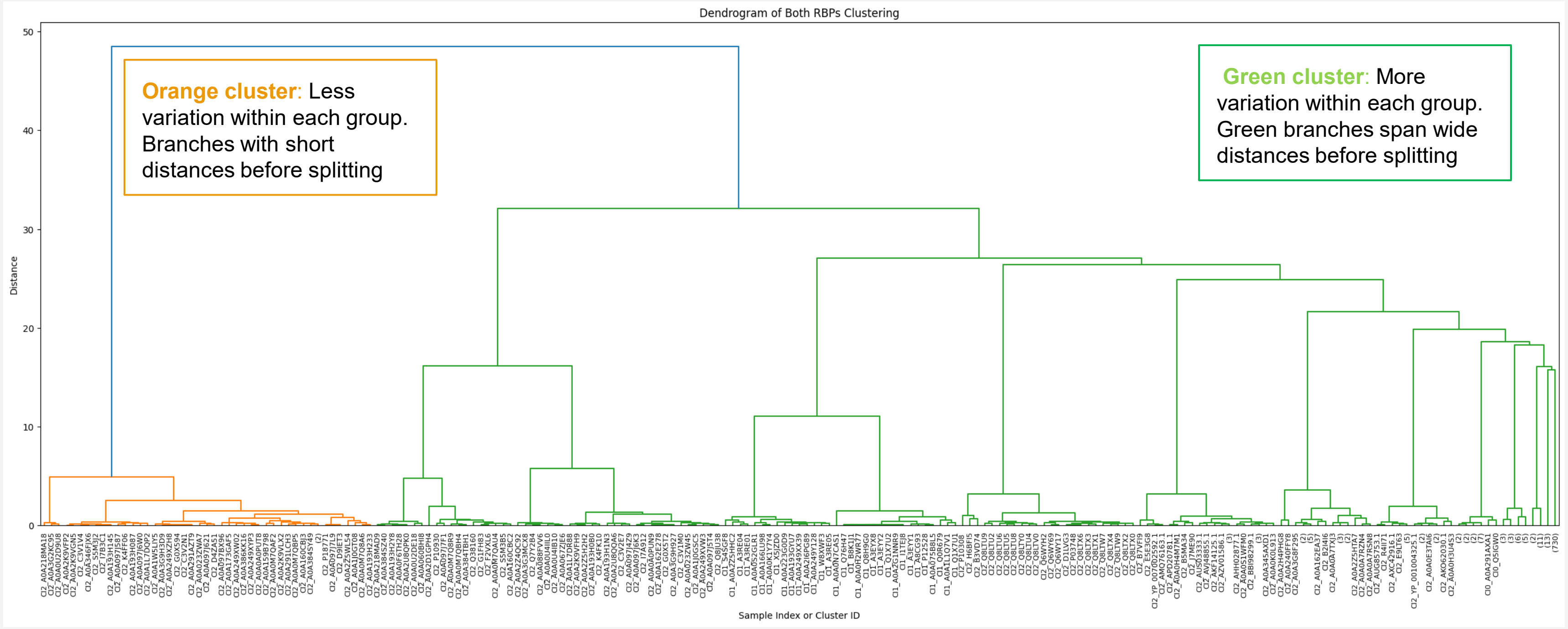
```python
best_score = -1
best_params = {}

for k in range(2, 10):
    for init_method in ['k-means++', 'random']:
        model = KMeans(n_clusters=k, init=init_method, random_state=42, n_init='auto')
        labels = model.fit_predict(X)
        score = silhouette_score(X, labels)
        if score > best_score:
            best_score = score
            best_params = {'n_clusters': k, 'init': init_method}

print(f"Best params: {best_params} with silhouette score: {best_score:.4f}")
```



Cluster 0:
Numerous points but overlaps with **Cluster 2**
Cluster 2:
Has most points → dominant cluster in this dataset
Cluster 1, Cluster 3:
Less populated and overlap → similarity in characteristics

Interpretation:
- Distinct clusters → Validated and Non-Validated RBPs may have different characteristics

3rd Dataset Analysis – Dendrogram



Orange cluster: Less variation within each group. Branches with short distances before splitting

Green cluster: More variation within each group. Green branches span wide distances before splitting

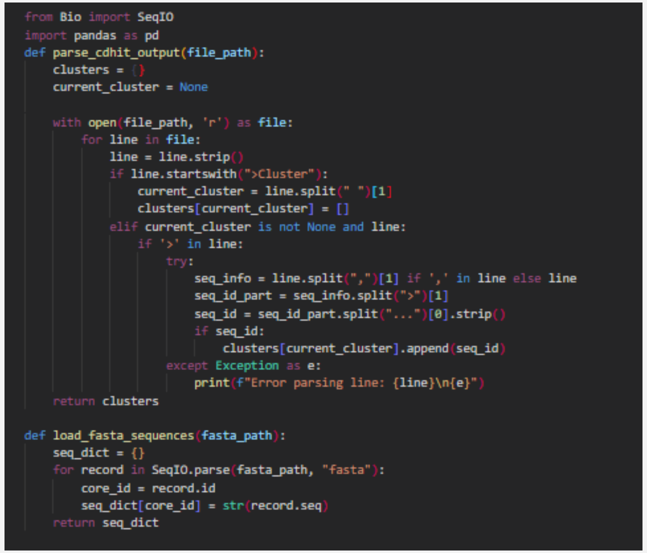
- Cluster shape: Broad, with higher internal variation
- Separation between clusters: High
- Cohesion between sub-groups
- Nº Clusters: Low

- Dissimilar sequences that share functional elements

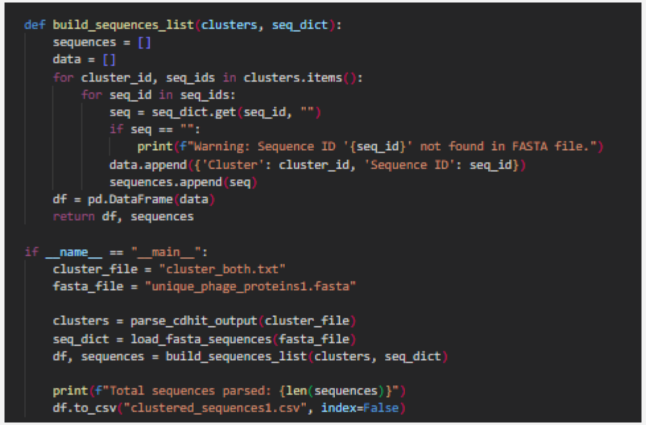
These clusters likely represent:

- Possibly novel or divergent RBPs
- Groupings of functionally analogous

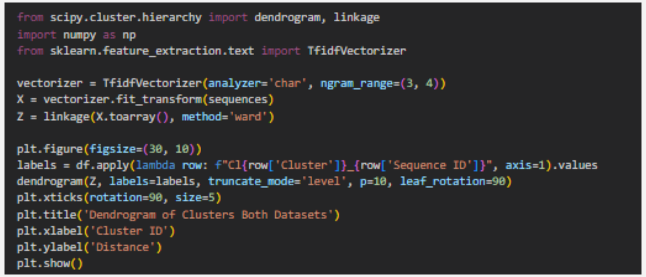
3rd Dataset – Both RBPs CD-HIT



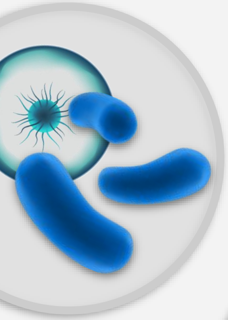
```python
from Bio import SeqIO
import pandas as pd
def parse_cdhit_output(file_path):
    clusters = {}
    current_cluster = None

    with open(file_path, 'r') as file:
        for line in file:
            line = line.strip()
            if line.startswith(">Cluster"):
                current_cluster = line.split(" ")[1]
                clusters[current_cluster] = []
            elif current_cluster is not None and line:
                if '>' in line:
                    try:
                        seq_info = line.split(",")[1] if ',' in line else line
                        seq_id_part = seq_info.split(">")[1]
                        seq_id = seq_id_part.split("...")[0].strip()
                        if seq_id:
                            clusters[current_cluster].append(seq_id)
                    except Exception as e:
                        print(f"Error parsing line: {line}\n{e}")
    return clusters

def load_fasta_sequences(fasta_path):
    seq_dict = {}
    for record in SeqIO.parse(fasta_path, "fasta"):
        core_id = record.id
        seq_dict[core_id] = str(record.seq)
    return seq_dict
```

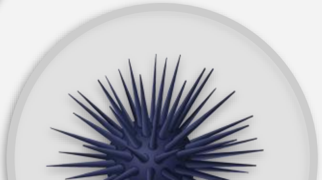


```python
def build_sequences_list(clusters, seq_dict):
    sequences = []
    data = []
    for cluster_id, seq_ids in clusters.items():
        for seq_id in seq_ids:
            seq = seq_dict.get(seq_id, "")
            if seq == "":
                print(f"Warning: Sequence ID '{seq_id}' not found in FASTA file.")
            data.append({'Cluster': cluster_id, 'Sequence ID': seq_id})
            sequences.append(seq)
    df = pd.DataFrame(data)
    return df, sequences

if __name__ == "__main__":
    cluster_file = "cluster_both.txt"
    fasta_file = "unique_phage_proteins1.fasta"

    clusters = parse_cdhit_output(cluster_file)
    seq_dict = load_fasta_sequences(fasta_file)
    df, sequences = build_sequences_list(clusters, seq_dict)

    print(f"Total sequences parsed: {len(sequences)}")
    df.to_csv("clustered_sequences1.csv", index=False)
```

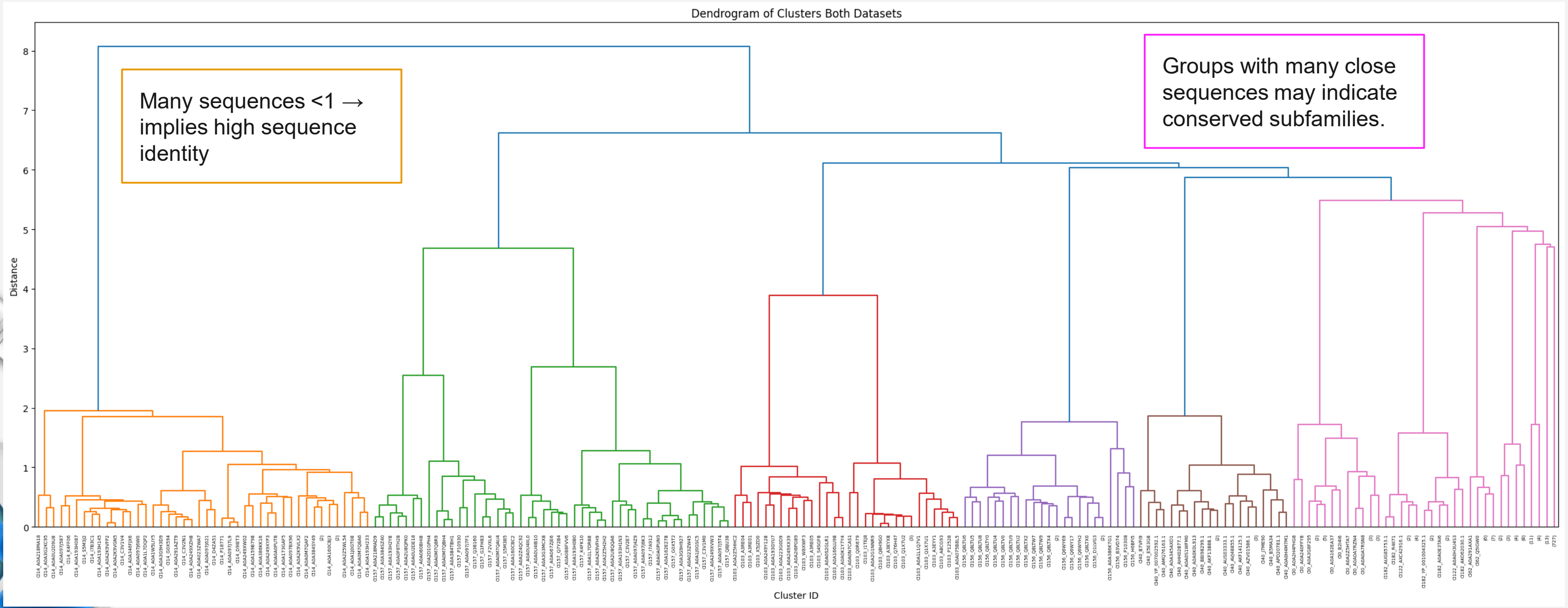


```python
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(analyzer='char', ngram_range=(3, 4))
X = vectorizer.fit_transform(sequences)
Z = linkage(X.toarray(), method='ward')

plt.figure(figsize=(30, 10))
labels = df.apply(lambda row: f"Cl{row['Cluster']}_{row['Sequence ID']}", axis=1).values
dendrogram(Z, labels=labels, truncate_mode='level', p=10, leaf_rotation=90)
plt.xticks(rotation=90, size=5)
plt.title('Dendrogram of Clusters Both Datasets')
plt.xlabel('Cluster ID')
plt.ylabel('Distance')
plt.show()
```





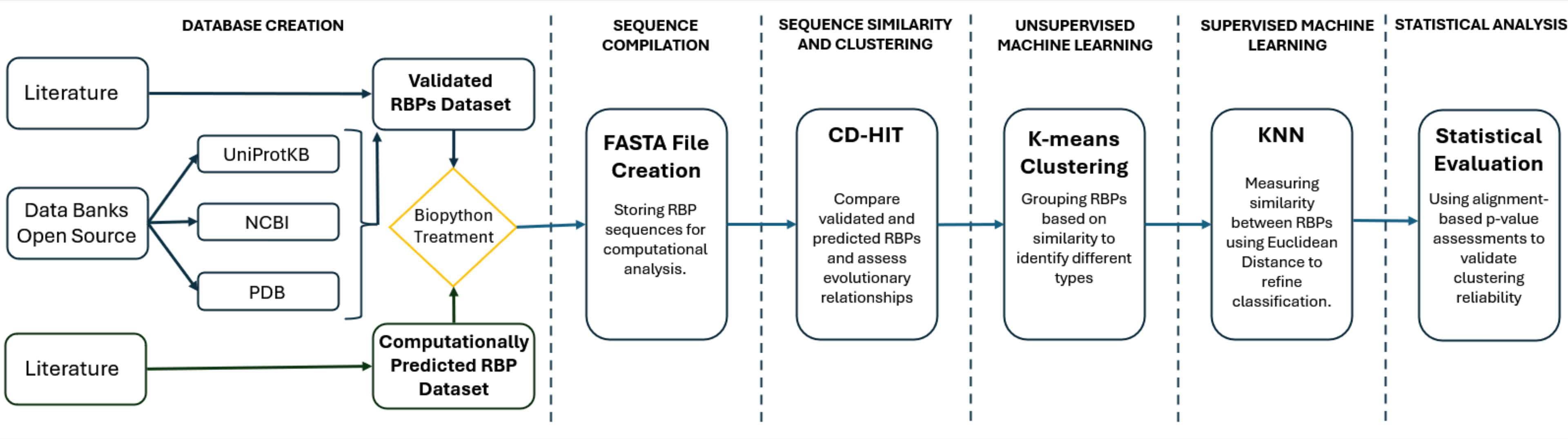
3rd Dataset – Both RBPs CD-HIT



- Cluster shape: Compact, narrow (short branches within)
- Separation between clusters: High (clear distinction)
- Intra-cluster similarity: High (identity ≥ 40%)
- Inter-cluster difference: High (hard cutoff)

Each cluster could correspond to:

- One phage RBP subtype,
- A known structural domain or tail fiber class,
- Evolutionary lineages with strong sequence conservation.


DATABASE CREATION
SEQUENCE COMPILATION
SEQUENCE SIMILARITY AND CLUSTERING
UNSUPERVISED MACHINE LEARNING
SUPERVISED MACHINE LEARNING
STATISTICAL ANALYSIS
Literature
Validated RBPs Dataset
UniProtKB
Data Banks Open Source
NCBI
PDB
Biopython Treatment
Literature
Computationally Predicted RBP Dataset
FASTA File Creation
Storing RBP sequences for computational analysis.
CD-HIT
Compare validated and predicted RBPs and assess evolutionary relationships
K-means Clustering
Grouping RBPs based on similarity to identify different types
KNN
Measuring similarity between RBPs using Euclidean Distance to refine classification.
Statistical Evaluation
Using alignment-based p-value assessments to validate clustering reliability

PROJECT WORKFLOW

05

Discussion

Discussion

Challenges

Creation of the provisory datasets:
- Filter and selection of RBPs from the 3 Databases
- Automatization of morphotype retrieval
- Datasets Uniformization

Validation of RBPs:
- Literature key word identification
- Explicit experimental data

Success

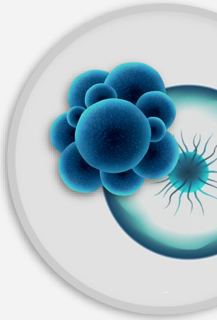
Curated RBP Database:
- Two datasets with Validated RBP and RBP candidates

Implemented dual clustering approaches:

- CD-HIT (identity-based): Captured conserved structural/evolutionary groups

- K-Means (motif-based): Revealed functional similarity based on k-mer features

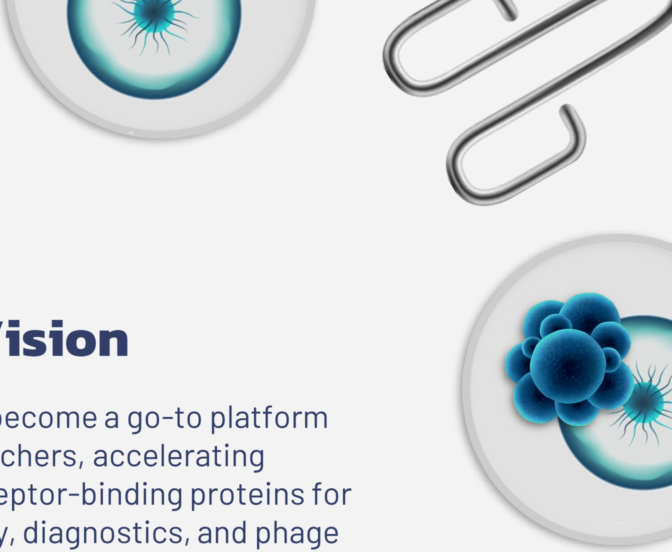
Visualized clustering via dendrograms:
- Assess sequence relationships



06

Conclusion

Conclusion



Next Steps

- **Propythia -** platform for the classification of biological sequences (proteins and DNA) using machine and deep learning

- Train classifiers using validated clusters as labels
- Deploy the RBP-ID web Based tool
- Enable user input and visualization, returning predicted RBP status, cluster info, and motifs
- Expand the validated dataset via literature mining

Vision

RBP-ID aims to become a go-to platform for phage researchers, accelerating discovery of receptor-binding proteins for synthetic biology, diagnostics, and phage therapy

Thanks