**Fig 5A: Gene Presence Absence Heat map**

1. **Custom Python script to convert strain-specific lists of resistome genes into a binary presence–absence matrix.**

import pandas as pd

def sort\_binary\_matrix(input\_file, output\_sorted\_file, output\_summary\_file):

# Read the input CSV file

df = pd.read\_csv(input\_file, index\_col=0)

# Sum the values in each row to use for sorting

df['sum'] = df.sum(axis=1)

# Sort the DataFrame based on the sum column and then drop the sum column

sorted\_df = df.sort\_values(by='sum', ascending=False).drop(columns='sum')

# Save the sorted matrix to a CSV file

sorted\_df.to\_csv(output\_sorted\_file)

# Create a summary DataFrame with Gene ID and count of strains where gene is present (value is 1)

gene\_summary = pd.DataFrame({

'Gene ID': sorted\_df.columns,

'Number of Strains with Gene Present': (sorted\_df == 1).sum(axis=0).values

})

# Save the summary DataFrame to a CSV file

gene\_summary.to\_csv(output\_summary\_file, index=False)

# Specify the working directory and file name

working\_directory = "C:/RGCB WORK Anagha/"

input\_file = working\_directory + "pamatrix\_output.csv"

output\_sorted\_file = working\_directory + "R\_sorted\_binary\_matrix.csv"

output\_summary\_file = working\_directory + "R\_gene\_presence\_summary.csv"

# Run the function with the file paths

sort\_binary\_matrix(input\_file, output\_sorted\_file, output\_summary\_file)

print(f"Files saved: {output\_sorted\_file}, {output\_summary\_file}")

1. **Custom Python script to sort the resulting matrix in descending order based on gene prevalence across strains, such that genes present in all strains were positioned at one end and those absent in all strains at the opposite end.**

import pandas as pd

import os

def generate\_binary\_matrix(input\_file, output\_file):

try:

# Read the input Excel file with the 'odf' engine

df = pd.read\_excel(input\_file, index\_col=0, engine='odf')

except ImportError:

raise ImportError("Missing optional dependency 'odf'. Install it using: pip install odfpy")

except Exception as e:

raise ValueError(f"Error reading input file: {e}")

# Initialize a set to gather unique gene IDs

gene\_ids = set()

# Iterate through each row to collect unique gene IDs

for genes\_list in df.values:

for gene\_group in genes\_list:

if isinstance(gene\_group, str):

# Assume genes are separated by commas, but allow flexibility

genes = [gene.strip() for gene in gene\_group.replace(';', ',').split(',')]

gene\_ids.update(genes)

# Convert the set of gene\_ids to a sorted list

gene\_ids = sorted(gene\_ids)

# Create a DataFrame for the binary matrix

binary\_matrix = pd.DataFrame(0, index=df.index, columns=gene\_ids)

# Fill the binary matrix with 1 if a gene is present in the corresponding row

for index, row in df.iterrows():

for gene\_group in row:

if isinstance(gene\_group, str):

genes = [gene.strip() for gene in gene\_group.replace(';', ',').split(',')]

for gene in genes:

if gene in binary\_matrix.columns:

binary\_matrix.at[index, gene] = 1

# Write the binary matrix to an output CSV file

binary\_matrix.to\_csv(output\_file)

# Set your working directory and file names

working\_directory = r'D:\RGCB WORK Anagha\TMC\Presence absence heatamap' # Update to your correct path

input\_file = os.path.join(working\_directory, 'pamatrix.xlsx') # assuming the file is Excel

output\_file = os.path.join(working\_directory, 'pamatrix\_output.csv')

# Generate the binary matrix

generate\_binary\_matrix(input\_file, output\_file)

print(f"Binary matrix saved to: {output\_file}")

1. **A custom R code was utilized to generate a gene presence absence heat map from the generated sorted presence absence matrix**

# Install and load required packages

install.packages("magrittr")

install.packages("pheatmap")

install.packages("readr")

library(pheatmap)

library(readr)

# Read the CSV file

data <- read\_csv("D:/RGCB WORK Anagha/TMC/Presence absence heatamap/R\_sorted\_binary\_matrix.csv")

# Convert the data frame to a matrix, excluding the first column if it contains row names

data\_matrix <- as.matrix(data[,-1])

# Optionally, set row names if the first column contains them

rownames(data\_matrix) <- data[[1]]

# Sort rows and columns by the sum of 1s

row\_order <- order(rowSums(data\_matrix), decreasing = TRUE)

col\_order <- order(colSums(data\_matrix), decreasing = TRUE)

sorted\_data\_matrix <- data\_matrix[row\_order, col\_order]

# Save the heatmap as a PDF

pdf("D:/RGCB WORK Anagha/TMC/Presence absence heatamap/heatmap\_output.pdf", width = 20, height = 10) # Adjust width and height as needed

# Generate the heatmap with uniform cell sizes and grid lines

pheatmap(sorted\_data\_matrix,

cluster\_rows = FALSE,

cluster\_cols = FALSE,

display\_numbers = FALSE,

color = colorRampPalette(c("lightyellow", "darkcyan"))(100),

border\_color = "white", # Set border color to black for uniform grid lines

fontsize = 5,

legend = FALSE, # Removes the color scale/legend

cellwidth = 10, # Set uniform cell width

cellheight = 10) # Set uniform cell height

dev.off() # Close the PDF device

**Fig 6A, B: Core Resistome and MLST Genes Maximum Likelihood Phylogenetic Trees**

1. **Concatenation of Core Resistome Genes:** We employed a custom Python script to concatenate all the individual resistance genes for each strain. The output is saved in the specified output folder.

import os

# Define input and output directories

input\_directory = r'E:\Anagha\_RGCB Work\Concatenation\Input'

output\_directory = r'E:\Anagha\_RGCB Work\Concatenation\Output'

# Ensure the output directory exists

os.makedirs(output\_directory, exist\_ok=True)

# Process each FASTA file in the input directory

for filename in os.listdir(input\_directory):

if filename.endswith('.fasta'):

input\_file\_path = os.path.join(input\_directory, filename)

output\_file\_path = os.path.join(output\_directory, f"{filename}")

# Read sequences from the input file

with open(input\_file\_path, 'r') as infile:

lines = infile.readlines()

# Initialize a list to store concatenated sequences

concatenated\_sequence = []

# Process the lines

for line in lines:

line = line.strip()

if not line.startswith('>'): # Skip headers

concatenated\_sequence.append(line)

# Join all sequences into a single sequence

concatenated\_sequence\_str = ''.join(concatenated\_sequence)

# Write the concatenated sequence to the output file

with open(output\_file\_path, 'w') as outfile:

outfile.write(concatenated\_sequence\_str)

print(f"Concatenated sequences (headers removed) written to {output\_file\_path}")

1. **Generation of ML Phylogenetic Tree using IQTree**

iqtree2 -s "inputfile.fasta" -m MFP -B 1000 -alrt 1000 -nt AUTO

The input file was analysed using IQTree’s ModelFinder Plus algorithm to find the best fit substitution model **(-m MFP)**. The generated tree was further validated using 1000 ultrafast bootstrap replicates **(-B 1000)** and 1000 SH-aLRT **(-alrt 1000)**.The number of threads was set automatically based on available CPU cores (-nt AUTO).

**Fig 7A: Drug Class Heat map:**

1. **A custom Python script was used to query an Excel-based database linking resistance genes to their corresponding antibiotic classes. For each strain, the script generated an output Excel file listing all antibiotic classes represented by the detected resistance genes.**

import pandas as pd

# Load the Excel file with HP tags and antibiotics

file\_path = r'C:\Users\Microbiome\HP\_tags\_and\_antibiotic\_list.xlsx'

xls = pd.ExcelFile(file\_path)

data = pd.read\_excel(xls, sheet\_name='Sheet1')

# Function to retrieve antibiotics for a given list of HP tags

def get\_antibiotics(hp\_tags, data):

# Create a DataFrame to maintain the order of HP tags

result\_df = pd.DataFrame({'HP\_Tag': hp\_tags})

result\_df['Antibiotics'] = 'Not Applicable' # Default value

# Filter the data for the given HP tags

filtered\_data = data[data['Gene'].isin(hp\_tags)]

# Update the result DataFrame with the corresponding antibiotics

for index, row in filtered\_data.iterrows():

tag = row['Gene']

antibiotics = row['Antibiotic'] # Assuming the column with antibiotics is named 'Antibiotic'

result\_df.loc[result\_df['HP\_Tag'] == tag, 'Antibiotics'] = antibiotics

return result\_df

# Latest list of HP tags for the strain

hp\_tags\_for\_strain = [

'HP\_1091', 'HP\_0935', 'HP\_0936', 'HP\_0969', 'HP\_0607', 'HP\_0019', 'HP\_0280', 'HP\_0166', 'HP\_1043', 'HP\_1021',

'HP\_0392', 'HP\_1067', 'HP\_0715', 'HP\_0748', 'HP\_0819', 'HP\_1198', 'HP\_0853', 'HP\_1428', 'HP\_0648', 'HP\_0501',

'HP\_1185', 'HP\_0301', 'HP\_0022', 'HP\_1205', 'HP\_0195', 'HP\_0703', 'HP\_0701', 'HP\_1565', 'HP\_0954', 'HP\_1181',

'HP\_0179', 'HP\_0619', 'HP\_0813', 'HP\_1220', 'HP\_0179', 'HP\_1576', 'HP\_1184', 'HP\_0606', 'HP\_1329', 'HP\_1082',

'HP\_0888', 'HP\_0250', 'HP\_1077', 'HP\_1465', 'HP\_1171', 'HP\_1466', 'HP\_0121', 'HP\_0355', 'HP\_1422', 'HP\_1153',

'HP\_1547', 'HP\_1232', 'HP\_0946', 'HP\_0480', 'HP\_1195', 'HP\_1048', 'HP\_0600', 'HP\_0738', 'HP\_0096', 'HP\_1365',

'HP\_0941', 'HP\_0475', 'HP\_0613', 'HP\_1206', 'HP\_1328', 'HP\_0558', 'HP\_0971', 'HP\_1035', 'HP\_1063', 'HP\_1081',

'HP\_1125', 'HP\_1197', 'HP\_1504', 'HP\_1464', 'HP\_0363', 'HP\_0393', 'HP\_0805', 'HP\_1556', 'HP\_0970', 'HP\_0190',

'HP\_1045', 'HP\_0683', 'HP\_1550', 'HP\_0705', 'HP\_0553', 'HP\_1087', 'HP\_1551', 'HP\_0561', 'HP\_0196', 'HP\_1165',

'HP\_1141', 'HP\_1431', 'HP\_0165', 'HP\_0597', 'HP\_0642', 'HP\_1434', 'HP\_0759', 'HP\_0164', 'HP\_0471', 'HP\_0616',

'HP\_1488', 'HP\_1442', 'HP\_0302', 'HP\_0244', 'HP\_1364', 'HP\_1375', 'HP\_0397'

]

# Get antibiotics for the provided strain

antibiotics\_for\_strain = get\_antibiotics(hp\_tags\_for\_strain, data)

# Display the results

print(antibiotics\_for\_strain)

# Optionally, save the result to a new Excel file

antibiotics\_for\_strain.to\_excel(r'C:\Users\Microbiome\antibiotics\_for\_strain.xlsx', index=False)

1. **Subsequently, a custom Python script was employed to quantify the number of resistance genes associated with each antibiotic class for every strain. The resulting antibiotic class count data was exported to a CSV file for further analysis.**

import pandas as pd

# Set the working directory path for the TMC110 file

file\_path = r'C:\RGCB WORK Anagha\TMC110.xlsx'

# Load the TMC110.xlsx file

tmc110\_df = pd.read\_excel(file\_path)

# List of antibiotics provided by you (maintain the order)

antibiotics\_list = [

'A201A', 'hygromycin A', 'iboxamycin', 'retapamulin', 'virginiamycin M1', 'acriflavine',

'norfloxacin', 'ciprofloxacin', 'amikacin', 'kanamycin A', 'novobiocin', 'tobramycin',

'azithromycin', 'erythromycin', 'penicillin', 'aztreonam', 'kitasamycin', 'rokitamycin',

'tetracycline', 'bacitracin', 'chloramphenicol', 'tedizolid', 'linezolid', 'florfenicol',

'mipenem', 'rifampin', 'ticarcillin', 'trimethoprim', 'thiamphenicol', 'chlortetracycline',

'demeclocycline', 'doxycycline', 'minocycline', 'oxytetracycline', 'gatifloxacin',

'levofloxacin', 'moxifloxacin', 'ofloxacin', 'clarithromycin', 'clindamycin', 'lincomycin',

'tiamulin', 'cloxacillin', 'oxacillin', 'colistin', 'dalfopristin', 'griseoviridin',

'madumycin II', 'defensin', 'polymyxin B', 'roxithromycin', 'fosfomycin', 'fusidic acid',

'gentamicin', 'isoniazid', 'nalidixic acid', 'norfloxacin', 'sparfloxacin', 'teicoplanin',

'mafenide', 'sulfacetamide', 'sulfadimidine', 'sulfadiazine', 'sulfisoxazole', 'methicillin',

'metronidazole', 'mupirocin', 'oleandomycin', 'oxytetracycline', 'pleuromutilin', 'pulvomycin',

'rifabutin', 'rifaximin', 'triclosan', 'tigecycline', 'vancomycin'

]

# Ensure the 'Antibiotics' column is cleaned of extra spaces or unwanted characters

tmc110\_df['Antibiotics'] = tmc110\_df['Antibiotics'].str.strip().str.lower()

# Dictionary to store the counts of antibiotics

antibiotic\_counts = {}

# Count occurrences of each antibiotic in the TMC110 file in the original order

for antibiotic in antibiotics\_list:

count = tmc110\_df['Antibiotics'].str.contains(antibiotic.lower(), case=False, na=False).sum()

antibiotic\_counts[antibiotic] = count

# Convert the result to a dataframe, ensuring the order is maintained

antibiotic\_counts\_df = pd.DataFrame(list(antibiotic\_counts.items()), columns=['Antibiotic', 'Count'])

# Save the output to a CSV file in the same working directory

antibiotic\_counts\_df.to\_csv(r'C:\RGCB WORK Anagha\antibiotic\_counts.csv', index=False)

# Display the result

print(antibiotic\_counts\_df)

1. **A Custom R script was used to generate a heat map visualizing the distribution of antibiotic classes across strains, based on the previously generated antibiotic class count data.**

# Install necessary package if not already installed

if (!require("pheatmap")) {

install.packages("pheatmap", dependencies = TRUE)

}

# Load the pheatmap package

library(pheatmap)

# Read the CSV file

data <- read.csv("D:/RGCB WORK Anagha/Heatmap Resistome.csv", row.names = 1)

# Convert the data into a matrix

data\_matrix <- as.matrix(data)

# Open a PDF device to save the heatmap

pdf("Heatmap\_Resistome\_No\_Clustering.pdf", width = 13, height = 25)

# Generate the heatmap without clustering

pheatmap(data\_matrix,

cluster\_rows = FALSE, # No clustering for rows

cluster\_cols = FALSE, # No clustering for columns

display\_numbers = TRUE, # Display the numbers in the cells

fontsize\_number = 7, # Adjust the font size of the numbers

number\_color = "black", # Change number color

cellwidth = 20, # Adjust cell width if needed

cellheight = 20, # Adjust cell height if needed

color = colorRampPalette(c("white", "maroon"))(50), # Color palette

main = "Heatmap of Antimicrobial Resistance Genes",

fontsize\_row = 12, # Font size for row labels

fontsize\_col = 12, # Font size for column labels

fontface\_row = "bold", # Make row labels bold

fontface\_col = "bold") # Make column labels bold

# Close the PDF device

dev.off()

**Supplementary Figure 1: Antibiotic Class Heat map**

1. **A custom Python script was developed to identify the antibiotics associated with the resistance genes detected in each strain. This was done by querying an Excel-based reference database that links each resistance gene to the corresponding antibiotic(s). For every strain, the script generated an output Excel file listing all antibiotics potentially impacted by the identified resistance genes.**

import pandas as pd

# Load the Excel file with HP tags and antibiotics

file\_path = r'C:\Users\Microbiome\HP\_tags\_and\_antibiotic\_list.xlsx'

xls = pd.ExcelFile(file\_path)

data = pd.read\_excel(xls, sheet\_name='Sheet1')

# Function to retrieve antibiotics for a given list of HP tags

def get\_antibiotics(hp\_tags, data):

# Create a DataFrame to maintain the order of HP tags

result\_df = pd.DataFrame({'HP\_Tag': hp\_tags})

result\_df['Antibiotics'] = 'Not Applicable' # Default value

# Filter the data for the given HP tags

filtered\_data = data[data['Gene'].isin(hp\_tags)]

# Update the result DataFrame with the corresponding antibiotics

for index, row in filtered\_data.iterrows():

tag = row['Gene']

antibiotics = row['Antibiotic'] # Assuming the column with antibiotics is named 'Antibiotic'

result\_df.loc[result\_df['HP\_Tag'] == tag, 'Antibiotics'] = antibiotics

return result\_df

# Latest list of HP tags for the strain

hp\_tags\_for\_strain = [

'HP\_1091', 'HP\_0935', 'HP\_0936', 'HP\_0969', 'HP\_0607', 'HP\_0019', 'HP\_0280', 'HP\_0166', 'HP\_1043', 'HP\_1021',

'HP\_0392', 'HP\_1067', 'HP\_0715', 'HP\_0748', 'HP\_0819', 'HP\_1198', 'HP\_0853', 'HP\_1428', 'HP\_0648', 'HP\_0501',

'HP\_1185', 'HP\_0301', 'HP\_0022', 'HP\_1205', 'HP\_0195', 'HP\_0703', 'HP\_0701', 'HP\_1565', 'HP\_0954', 'HP\_1181',

'HP\_0179', 'HP\_0619', 'HP\_0813', 'HP\_1220', 'HP\_0179', 'HP\_1576', 'HP\_1184', 'HP\_0606', 'HP\_1329', 'HP\_1082',

'HP\_0888', 'HP\_0250', 'HP\_1077', 'HP\_1465', 'HP\_1171', 'HP\_1466', 'HP\_0121', 'HP\_0355', 'HP\_1422', 'HP\_1153',

'HP\_1547', 'HP\_1232', 'HP\_0946', 'HP\_0480', 'HP\_1195', 'HP\_1048', 'HP\_0600', 'HP\_0738', 'HP\_0096', 'HP\_1365',

'HP\_0941', 'HP\_0475', 'HP\_0613', 'HP\_1206', 'HP\_1328', 'HP\_0558', 'HP\_0971', 'HP\_1035', 'HP\_1063', 'HP\_1081',

'HP\_1125', 'HP\_1197', 'HP\_1504', 'HP\_1464', 'HP\_0363', 'HP\_0393', 'HP\_0805', 'HP\_1556', 'HP\_0970', 'HP\_0190',

'HP\_1045', 'HP\_0683', 'HP\_1550', 'HP\_0705', 'HP\_0553', 'HP\_1087', 'HP\_1551', 'HP\_0561', 'HP\_0196', 'HP\_1165',

'HP\_1141', 'HP\_1431', 'HP\_0165', 'HP\_0597', 'HP\_0642', 'HP\_1434', 'HP\_0759', 'HP\_0164', 'HP\_0471', 'HP\_0616',

'HP\_1488', 'HP\_1442', 'HP\_0302', 'HP\_0244', 'HP\_1364', 'HP\_1375', 'HP\_0397'

]

# Get antibiotics for the provided strain

antibiotics\_for\_strain = get\_antibiotics(hp\_tags\_for\_strain, data)

# Display the results

print(antibiotics\_for\_strain)

# Optionally, save the result to a new Excel file

antibiotics\_for\_strain.to\_excel(r'C:\Users\Microbiome\antibiotics\_for\_strain.xlsx', index=False)

1. **A custom Python script was then used to calculate the number of resistance genes linked to each antibiotic for every strain. The resulting data, representing antibiotic-wise gene counts per strain, was exported as a CSV file.**

import pandas as pd

# Set the working directory path for the TMC110 file

file\_path = r'C:\RGCB WORK Anagha\TMC110.xlsx'

# Load the TMC110.xlsx file

tmc110\_df = pd.read\_excel(file\_path)

# List of antibiotics provided by you (maintain the order)

antibiotics\_list = [

'A201A', 'hygromycin A', 'iboxamycin', 'retapamulin', 'virginiamycin M1', 'acriflavine',

'norfloxacin', 'ciprofloxacin', 'amikacin', 'kanamycin A', 'novobiocin', 'tobramycin',

'azithromycin', 'erythromycin', 'penicillin', 'aztreonam', 'kitasamycin', 'rokitamycin',

'tetracycline', 'bacitracin', 'chloramphenicol', 'tedizolid', 'linezolid', 'florfenicol',

'mipenem', 'rifampin', 'ticarcillin', 'trimethoprim', 'thiamphenicol', 'chlortetracycline',

'demeclocycline', 'doxycycline', 'minocycline', 'oxytetracycline', 'gatifloxacin',

'levofloxacin', 'moxifloxacin', 'ofloxacin', 'clarithromycin', 'clindamycin', 'lincomycin',

'tiamulin', 'cloxacillin', 'oxacillin', 'colistin', 'dalfopristin', 'griseoviridin',

'madumycin II', 'defensin', 'polymyxin B', 'roxithromycin', 'fosfomycin', 'fusidic acid',

'gentamicin', 'isoniazid', 'nalidixic acid', 'norfloxacin', 'sparfloxacin', 'teicoplanin',

'mafenide', 'sulfacetamide', 'sulfadimidine', 'sulfadiazine', 'sulfisoxazole', 'methicillin',

'metronidazole', 'mupirocin', 'oleandomycin', 'oxytetracycline', 'pleuromutilin', 'pulvomycin',

'rifabutin', 'rifaximin', 'triclosan', 'tigecycline', 'vancomycin'

]

# Ensure the 'Antibiotics' column is cleaned of extra spaces or unwanted characters

tmc110\_df['Antibiotics'] = tmc110\_df['Antibiotics'].str.strip().str.lower()

# Dictionary to store the counts of antibiotics

antibiotic\_counts = {}

# Count occurrences of each antibiotic in the TMC110 file in the original order

for antibiotic in antibiotics\_list:

count = tmc110\_df['Antibiotics'].str.contains(antibiotic.lower(), case=False, na=False).sum()

antibiotic\_counts[antibiotic] = count

# Convert the result to a dataframe, ensuring the order is maintained

antibiotic\_counts\_df = pd.DataFrame(list(antibiotic\_counts.items()), columns=['Antibiotic', 'Count'])

# Save the output to a CSV file in the same working directory

antibiotic\_counts\_df.to\_csv(r'C:\RGCB WORK Anagha\antibiotic\_counts.csv', index=False)

# Display the result

print(antibiotic\_counts\_df)

1. **A custom R script was used to generate a heat map illustrating the distribution of ARGs associated with antibiotics across strains, based on the previously generated count data.**

# Install necessary package if not already installed

if (!require("pheatmap")) {

install.packages("pheatmap", dependencies = TRUE)

}

# Load the pheatmap package

library(pheatmap)

# Read the CSV file

data <- read.csv("D:/RGCB WORK Anagha/Heatmap Resistome.csv", row.names = 1)

# Convert the data into a matrix

data\_matrix <- as.matrix(data)

# Open a PDF device to save the heatmap

pdf("Heatmap\_Resistome\_No\_Clustering.pdf", width = 13, height = 25)

# Generate the heatmap without clustering

pheatmap(data\_matrix,

cluster\_rows = FALSE, # No clustering for rows

cluster\_cols = FALSE, # No clustering for columns

display\_numbers = TRUE, # Display the numbers in the cells

fontsize\_number = 7, # Adjust the font size of the numbers

number\_color = "black", # Change number color

cellwidth = 20, # Adjust cell width if needed

cellheight = 20, # Adjust cell height if needed

color = colorRampPalette(c("white", "maroon"))(50), # Color palette

main = "Heatmap of Antimicrobial Resistance Genes",

fontsize\_row = 12, # Font size for row labels

fontsize\_col = 12, # Font size for column labels

fontface\_row = "bold", # Make row labels bold

fontface\_col = "bold") # Make column labels bold

# Close the PDF device

dev.off()