Proxy Analysis

import pandas as pd

# Load the Diatoms data

diatoms\_data = pd.read\_excel('/mnt/data/Diatoms.xlsx')

# Display the first few rows and summary of the Diatoms data

diatoms\_summary = diatoms\_data.describe(include='all')

diatoms\_data.head(), diatoms\_summary

# Load the Cladocera data

cladocera\_data = pd.read\_excel('/mnt/data/Cladocera.xlsx')

# Display the first few rows and summary of the Cladocera data

cladocera\_summary = cladocera\_data.describe(include='all')

cladocera\_data.head(), cladocera\_summary

# Load the Pediastrum data with the correct delimiter and convert commas to dots

pediastrum\_data = pd.read\_csv('/mnt/data/Pediastrum.csv', delimiter=';', decimal=',')

# Display the first few rows and summary of the Pediastrum data

pediastrum\_summary = pediastrum\_data.describe(include='all')

pediastrum\_data.head(), pediastrum\_summary

# Load the Ostracode data

ostracode\_data = pd.read\_excel('/mnt/data/Ostracode.xlsx')

# Display the first few rows and summary of the Ostracode data

ostracode\_summary = ostracode\_data.describe(include='all')

ostracode\_data.head(), ostracode\_summary

# multilayer network analysis (intralayer network)

import networkx as nx

import matplotlib.pyplot as plt

# Create intralayer networks

def create\_intralayer\_network(interaction\_matrices, community\_name):

G = nx.Graph()

for period, interaction\_matrix in interaction\_matrices.items():

for i in range(len(interaction\_matrix)):

for j in range(i+1, len(interaction\_matrix)):

if interaction\_matrix.iloc[i, j] > 0.5:

G.add\_edge((community\_name, period, interaction\_matrix.index[i]),

(community\_name, period, interaction\_matrix.columns[j]),

weight=interaction\_matrix.iloc[i, j])

return G

G\_pediastrum = create\_intralayer\_network(interaction\_matrices\_pediastrum, 'Pediastrum')

G\_ostracode = create\_intralayer\_network(interaction\_matrices\_ostracode, 'Ostracode')

G\_cladocera = create\_intralayer\_network(interaction\_matrices\_cladocera, 'Cladocera')

G\_diatom = create\_intralayer\_network(interaction\_matrices\_diatom, 'Diatom')

# Combine all graphs into a single multilayer network

multilayer\_network = nx.compose\_all([G\_pediastrum, G\_ostracode, G\_cladocera, G\_diatom])

# Add interlayer edges based on correlations

def add\_interlayer\_edges(data1, data2, name1, name2, period):

period\_data1 = data1[data1['Climate Period'] == period].drop(columns=['Climate Period', 'Age cal yr BP', 'Age CE'])

period\_data2 = data2[data2['Climate Period'] == period].drop(columns=['Climate Period', 'Age cal yr BP', 'Age CE'])

common\_columns = period\_data1.columns.intersection(period\_data2.columns)

for col in common\_columns:

correlation = period\_data1[col].corr(period\_data2[col])

if correlation > 0.5:

multilayer\_network.add\_edge((name1, period, col), (name2, period, col), weight=correlation)

for period in climate\_periods:

add\_interlayer\_edges(pediastrum\_data, ostracode\_data, 'Pediastrum', 'Ostracode', period)

add\_interlayer\_edges(pediastrum\_data, cladocera\_data, 'Pediastrum', 'Cladocera', period)

add\_interlayer\_edges(pediastrum\_data, diatom\_data, 'Pediastrum', 'Diatom', period)

add\_interlayer\_edges(ostracode\_data, cladocera\_data, 'Ostracode', 'Cladocera', period)

add\_interlayer\_edges(ostracode\_data, diatom\_data, 'Ostracode', 'Diatom', period)

add\_interlayer\_edges(cladocera\_data, diatom\_data, 'Cladocera', 'Diatom', period)

# Analyze the Multilayer Network

num\_layers = len(climate\_periods)

num\_nodes = multilayer\_network.number\_of\_nodes()

num\_edges = multilayer\_network.number\_of\_edges()

# Visualize the Multilayer Network

pos = nx.spring\_layout(multilayer\_network)

plt.figure(figsize=(12, 12))

nx.draw(multilayer\_network, pos, with\_labels=True, node\_size=50, font\_size=8)

plt.title("Multilayer Network of Biological Communities Across Climate Periods")

plt.show()

num\_layers, num\_nodes, num\_edges

#lets execute again

import numpy as np

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

# Define the climate periods

climate\_periods = {

"Roman Warming Period": (0, 200),

"Dark Age Cold Period": (200, 550),

"Sui and Tang Dynasties Warm Period": (550, 700),

"Medieval Climate Anomaly": (900, 1400),

"Little Ice Age": (1400, 1850),

"Current Warming Period": (1850, 2024)

}

# Function to add climate period column

def add\_climate\_period\_column(df):

def get\_climate\_period(age\_ce):

for period, (start, end) in climate\_periods.items():

if start <= age\_ce <= end:

return period

return np.nan

df['Climate Period'] = df['Age CE'].apply(get\_climate\_period)

return df

# Function to prepare interaction matrices

def prepare\_interaction\_matrices(df, climate\_periods):

interaction\_matrices = {}

for period in climate\_periods:

period\_data = df[df['Climate Period'] == period].drop(columns=['Climate Period', 'Age cal yr BP', 'Age CE'])

if not period\_data.empty:

correlation\_matrix = period\_data.corr()

interaction\_matrices[period] = correlation\_matrix

return interaction\_matrices

# Load the data

pediastrum\_data = add\_climate\_period\_column(pd.read\_csv('/mnt/data/Pediastrum.csv', delimiter=';', decimal=','))

ostracode\_data = add\_climate\_period\_column(pd.read\_excel('/mnt/data/Ostracode.xlsx'))

cladocera\_data = add\_climate\_period\_column(pd.read\_excel('/mnt/data/Cladocera.xlsx'))

diatom\_data = add\_climate\_period\_column(pd.read\_excel('/mnt/data/Diatoms.xlsx'))

# Prepare interaction matrices

interaction\_matrices\_pediastrum = prepare\_interaction\_matrices(pediastrum\_data, climate\_periods)

interaction\_matrices\_ostracode = prepare\_interaction\_matrices(ostracode\_data, climate\_periods)

interaction\_matrices\_cladocera = prepare\_interaction\_matrices(cladocera\_data, climate\_periods)

interaction\_matrices\_diatom = prepare\_interaction\_matrices(diatom\_data, climate\_periods)

# Create intralayer networks

def create\_intralayer\_network(interaction\_matrices, community\_name):

G = nx.Graph()

for period, interaction\_matrix in interaction\_matrices.items():

for i in range(len(interaction\_matrix)):

for j in range(i+1, len(interaction\_matrix)):

if interaction\_matrix.iloc[i, j] > 0.5:

G.add\_edge((community\_name, period, interaction\_matrix.index[i]),

(community\_name, period, interaction\_matrix.columns[j]),

weight=interaction\_matrix.iloc[i, j])

return G

G\_pediastrum = create\_intralayer\_network(interaction\_matrices\_pediastrum, 'Pediastrum')

G\_ostracode = create\_intralayer\_network(interaction\_matrices\_ostracode, 'Ostracode')

G\_cladocera = create\_intralayer\_network(interaction\_matrices\_cladocera, 'Cladocera')

G\_diatom = create\_intralayer\_network(interaction\_matrices\_diatom, 'Diatom')

# Combine all graphs into a single multilayer network

multilayer\_network = nx.compose\_all([G\_pediastrum, G\_ostracode, G\_cladocera, G\_diatom])

# Add interlayer edges based on correlations

def add\_interlayer\_edges(data1, data2, name1, name2, period):

period\_data1 = data1[data1['Climate Period'] == period].drop(columns=['Climate Period', 'Age cal yr BP', 'Age CE'])

period\_data2 = data2[data2['Climate Period'] == period].drop(columns=['Climate Period', 'Age cal yr BP', 'Age CE'])

common\_columns = period\_data1.columns.intersection(period\_data2.columns)

for col in common\_columns:

correlation = period\_data1[col].corr(period\_data2[col])

if correlation > 0.5:

multilayer\_network.add\_edge((name1, period, col), (name2, period, col), weight=correlation)

for period in climate\_periods:

add\_interlayer\_edges(pediastrum\_data, ostracode\_data, 'Pediastrum', 'Ostracode', period)

add\_interlayer\_edges(pediastrum\_data, cladocera\_data, 'Pediastrum', 'Cladocera', period)

add\_interlayer\_edges(pediastrum\_data, diatom\_data, 'Pediastrum', 'Diatom', period)

add\_interlayer\_edges(ostracode\_data, cladocera\_data, 'Ostracode', 'Cladocera', period)

add\_interlayer\_edges(ostracode\_data, diatom\_data, 'Ostracode', 'Diatom', period)

add\_interlayer\_edges(cladocera\_data, diatom\_data, 'Cladocera', 'Diatom', period)

# Analyze the Multilayer Network

num\_layers = len(climate\_periods)

num\_nodes = multilayer\_network.number\_of\_nodes()

num\_edges = multilayer\_network.number\_of\_edges()

# Visualize the Multilayer Network

plt.figure(figsize=(12, 12))

pos = nx.spring\_layout(multilayer\_network, k=0.1) # Adjust k for layout spacing

nx.draw(multilayer\_network, pos, with\_labels=True, node\_size=50, font\_size=8)

plt.title("Multilayer Network of Biological Communities Across Climate Periods")

plt.show()

num\_layers, num\_nodes, num\_edges

# visualization for multilayer network analysis

# Assign colors to each community for better distinction

community\_colors = {

'Pediastrum': 'red',

'Ostracode': 'blue',

'Cladocera': 'green',

'Diatom': 'orange'

}

# Assign positions for each layer to separate them vertically

layer\_positions = {period: index for index, period in enumerate(climate\_periods)}

# Create positions dictionary for networkx

pos = {}

for node in multilayer\_network.nodes():

community, period, species = node

pos[node] = (layer\_positions[period], np.random.rand() + layer\_positions[period] \* 10)

# Draw the network

plt.figure(figsize=(15, 15))

for community, color in community\_colors.items():

community\_nodes = [node for node in multilayer\_network.nodes() if node[0] == community]

nx.draw\_networkx\_nodes(multilayer\_network, pos, nodelist=community\_nodes, node\_color=color, label=community, node\_size=50)

nx.draw\_networkx\_edges(multilayer\_network, pos, alpha=0.3)

nx.draw\_networkx\_labels(multilayer\_network, pos, font\_size=7)

plt.legend()

plt.title("Multilayer Network of Biological Communities Across Climate Periods")

plt.xlabel('Climate Periods (Vertical Separation)')

plt.ylabel('Species Interactions (Horizontal Separation)')

plt.grid(True)

plt.show()