ROC Analysis

Data Preprocessing:

Load your palaeoecological data (e.g., pollen and diatom counts).

Transform the data into relative abundances if needed.

Ensure your data has a consistent time axis.

Calculate Dissimilarity:

Use a suitable dissimilarity measure (e.g., chi-squared, Bray-Curtis).

Calculate dissimilarity between successive time points.

Smoothing and Rescaling:

Apply smoothing techniques (e.g., moving average).

Rescale the dissimilarity scores to the desired time interval (e.g., per 50 years).

Random Sampling and Iteration:

Implement random sampling to account for variability.

Repeat calculations multiple times (e.g., 999 iterations) to generate a distribution of RoC scores.

Identify Significant Peaks:

Determine significant RoC peaks by comparing with a threshold (e.g., 95th percentile of RoC scores).

Code

import numpy as np

import pandas as pd

from scipy.spatial.distance import cdist

from scipy.ndimage import uniform\_filter1d

# Load data

data = pd.read\_csv('your\_data.csv') # Replace with your data source

ages = data['Age']

pollen\_counts = data.iloc[:, 1:] # Adjust to select the correct columns

# Calculate relative abundances

relative\_abundances = pollen\_counts.div(pollen\_counts.sum(axis=1), axis=0)

# Calculate dissimilarity (using chi-squared as an example)

def chi\_squared\_dissimilarity(p1, p2):

return 0.5 \* np.sum((p1 - p2) \*\* 2 / (p1 + p2 + 1e-10))

dissimilarities = [chi\_squared\_dissimilarity(relative\_abundances.iloc[i], relative\_abundances.iloc[i+1])

for i in range(len(relative\_abundances)-1)]

# Apply smoothing (e.g., moving average)

smoothed\_dissimilarities = uniform\_filter1d(dissimilarities, size=14)

# Rescale to RoC per 50 years (example)

time\_intervals = np.diff(ages)

roc\_scores = smoothed\_dissimilarities / time\_intervals

# Random sampling and iteration

iterations = 999

all\_roc\_scores = []

for \_ in range(iterations):

sampled\_indices = np.random.choice(range(len(roc\_scores)), size=len(roc\_scores), replace=True)

sampled\_roc\_scores = roc\_scores[sampled\_indices]

all\_roc\_scores.append(sampled\_roc\_scores)

all\_roc\_scores = np.array(all\_roc\_scores)

roc\_95th\_quantile = np.percentile(all\_roc\_scores, 95, axis=0)

# Identify significant RoC peaks

significant\_peaks = roc\_scores > roc\_95th\_quantile

# Output results

results = pd.DataFrame({

'Age': ages[1:], # Adjust for the correct age array

'RoC': roc\_scores,

'Significant': significant\_peaks

})

print(results)