import healpy as hp

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import pearsonr

# --- Load polarization maps ---

q\_map = hp.read\_map("COM\_CMB\_IQU-smica\_2048\_R3.00\_oe2.fits", field=1)  # Q (E-mode)

u\_map = hp.read\_map("COM\_CMB\_IQU-smica\_2048\_R3.00\_oe2.fits", field=2)  # U (B-mode)

nside = hp.get\_nside(q\_map)

npix = hp.nside2npix(nside)

# --- Cold spot dispatch root ---

theta\_ref = np.radians(90 - 57)

phi\_ref = np.radians(207)

vec\_ref = hp.ang2vec(theta\_ref, phi\_ref)

# --- Ring profile extractor ---

def ring\_profile(field, vec\_center, num\_rings=10, max\_radius\_deg=50):

    vecs = np.array(hp.pix2vec(nside, np.arange(npix)))

    dots = np.dot(vec\_center, vecs)

    dots = np.clip(dots, -1.0, 1.0)

    angs\_deg = np.degrees(np.arccos(dots))

    edges = np.linspace(0, max\_radius\_deg, num\_rings + 1)

    profile = []

    for i in range(num\_rings):

        mask = (angs\_deg >= edges[i]) & (angs\_deg < edges[i+1])

        profile.append(np.mean(field[mask]) if np.any(mask) else 0.0)

    return profile

# --- Reference Q/U ring profiles ---

q\_profile\_ref = ring\_profile(q\_map, vec\_ref)

u\_profile\_ref = ring\_profile(u\_map, vec\_ref)

# --- Curvature-weighted MBT shell ---

vecs = np.array(hp.pix2vec(nside, np.arange(npix)))

dots = np.dot(vec\_ref, vecs)

dots = np.clip(dots, -1.0, 1.0)

angs\_rad = np.arccos(dots)

height = 1.0

steepness = 3.0

curvature\_values = height / (1 + steepness \* angs\_rad\*\*2)

curvature\_values /= np.max(curvature\_values)

mbt\_field = curvature\_values

mbt\_profile\_ref = ring\_profile(mbt\_field, vec\_ref)

# --- Reference correlations ---

corr\_q\_ref, \_ = pearsonr(q\_profile\_ref, mbt\_profile\_ref)

corr\_u\_ref, \_ = pearsonr(u\_profile\_ref, mbt\_profile\_ref)

print(f"🔍 MBT–Q Correlation: {corr\_q\_ref:.4f}")

print(f"🔍 MBT–U Correlation: {corr\_u\_ref:.4f}")

# --- Monte Carlo trials ---

num\_trials = 1000

correlations\_q, correlations\_u = [], []

print("⚙️ Running randomized MBT shell trials...")

for i in range(num\_trials):

    theta\_rand = np.radians(np.random.uniform(0, 180))

    phi\_rand = np.radians(np.random.uniform(0, 360))

    vec\_rand = hp.ang2vec(theta\_rand, phi\_rand)

    dots\_rand = np.dot(vec\_rand, vecs)

    dots\_rand = np.clip(dots\_rand, -1.0, 1.0)

    angs\_rand = np.arccos(dots\_rand)

    mbt\_rand = height / (1 + steepness \* angs\_rand\*\*2)

    mbt\_rand /= np.max(mbt\_rand)

    mbt\_profile\_rand = ring\_profile(mbt\_rand, vec\_ref)

    if np.std(mbt\_profile\_rand) == 0:

        continue

    r\_q, \_ = pearsonr(q\_profile\_ref, mbt\_profile\_rand)

    r\_u, \_ = pearsonr(u\_profile\_ref, mbt\_profile\_rand)

    correlations\_q.append(r\_q)

    correlations\_u.append(r\_u)

    if i % 50 == 0:

        print(f"Trial {i}/{num\_trials}…")

# --- Histograms ---

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.hist(correlations\_q, bins=50, color='deepskyblue', edgecolor='black')

plt.axvline(corr\_q\_ref, color='red', linestyle='--', label=f'MBT Q (r = {corr\_q\_ref:.2f})')

plt.title("Q Polarization Tail Correlation")

plt.xlabel("Pearson r"); plt.ylabel("Trials")

plt.grid(True, alpha=0.3); plt.legend()

plt.subplot(1, 2, 2)

plt.hist(correlations\_u, bins=50, color='slategray', edgecolor='black')

plt.axvline(corr\_u\_ref, color='red', linestyle='--', label=f'MBT U (r = {corr\_u\_ref:.2f})')

plt.title("U Polarization Tail Correlation")

plt.xlabel("Pearson r"); plt.ylabel("Trials")

plt.grid(True, alpha=0.3); plt.legend()

plt.tight\_layout()

plt.show()

# --- Tail stats ---

tail\_q = sum(r <= corr\_q\_ref for r in correlations\_q)

tail\_u = sum(r <= corr\_u\_ref for r in correlations\_u)

print(f"\n📈 MBT–Q tail hits: {tail\_q}/{len(correlations\_q)} → p = {tail\_q / len(correlations\_q):.4f}")

print(f"📈 MBT–U tail hits: {tail\_u}/{len(correlations\_u)} → p = {tail\_u / len(correlations\_u):.4f}")