Додаток 1

Лістинг Python-скрипта для обчислення Local Moran (LISA) з перестановками + карти кластерів і діаграми розсіяння Морана в розрізі територіальних громад

# Обчислення Local Moran (LISA) з перестановками + карти кластерів і діаграми розсіяння Морана

import numpy as np, pandas as pd, matplotlib.pyplot as plt

from matplotlib.patches import Polygon

from matplotlib.collections import PatchCollection

from PIL import Image, ImageDraw, ImageFont

from pathlib import Path

import shapefile

BASE = Path("/mnt/data")

YEAR = 2024

# --- вхідні заготовки з попередньої комірки ---

sf = shapefile.Reader(str(BASE / "IF\_reg\_TG\_bou\_7.shp"))

shapes = sf.shapes()

fields = [f[0] for f in sf.fields if f[0] != "DeletionFlag"]

recs = [dict(zip(fields, r)) for r in sf.records()]

code\_field = "katotth" if "katotth" in fields else ("katottg" if "katottg" in fields else None)

name\_field = "name"

codes = [r[code\_field] for r in recs]

names = [r[name\_field] for r in recs]

def poly\_centroid(points):

if len(points) < 3:

xs = [p[0] for p in points]; ys = [p[1] for p in points]

return (sum(xs)/len(xs), sum(ys)/len(ys))

a = 0.0; cx = 0.0; cy = 0.0

for i in range(len(points)):

x1,y1 = points[i]

x2,y2 = points[(i+1)%len(points)]

cross = x1\*y2 - x2\*y1

a += cross; cx += (x1 + x2)\*cross; cy += (y1 + y2)\*cross

a \*= 0.5

if abs(a) < 1e-12:

xs = [p[0] for p in points]; ys = [p[1] for p in points]

return (sum(xs)/len(xs), sum(ys)/len(ys))

cx /= (6\*a); cy /= (6\*a)

return (cx, cy)

centroids = []

patches = []

for shp in shapes:

pts = shp.points

parts = list(shp.parts) + [len(pts)]

if len(parts) >= 2:

ring = pts[parts[0]:parts[1]]

else:

ring = pts

centroids.append(poly\_centroid(ring))

patches.append(Polygon(ring, closed=True))

XY = np.array(centroids)

n = XY.shape[0]

def kNN\_weights(XY, k=5):

n = XY.shape[0]

W = np.zeros((n,n), dtype=float)

for i in range(n):

d = np.sqrt(np.sum((XY - XY[i])\*\*2, axis=1))

idx = np.argsort(d)

neigh = [j for j in idx if j!=i][:k]

for j in neigh:

W[i,j] = 1.0

row\_sums = W.sum(axis=1, keepdims=True)

row\_sums[row\_sums==0] = 1.0

return W / row\_sums

W = kNN\_weights(XY, k=5)

# --- підтягуємо grid з попередньої комірки (якщо втрачено — реконструюємо швидко) ---

import pandas as pd

tour = pd.read\_csv(BASE / "згруповано\_турзбір.csv")

tn\_2024 = tour[tour["Рік"]==YEAR].groupby("Код\_громади")["Всього\_туристо\_діб"].sum().rename("TN").reset\_index()

tn\_2024 = tn\_2024.rename(columns={"Код\_громади":"katottg"})

voda = pd.read\_excel(BASE / "voda.xlsx")

voda["sumW"] = voda[["R7","R8","R8\_1","R8\_2"]].fillna(0).sum(axis=1)

w\_2024 = voda[voda["PERIOD\_YEAR"]==YEAR].groupby("TG")["sumW"].sum().rename("W").reset\_index().rename(columns={"TG":"katottg"})

def agg\_ecol(path):

df = pd.read\_excel(path)

df["year"] = pd.to\_numeric(df["P\_YEAR"], errors="coerce")

d = df[df["year"]==YEAR].groupby("HKATOTTG")["POLLUTION\_VOL"].sum().rename("val").reset\_index()

return d.rename(columns={"HKATOTTG":"katottg"})

air = agg\_ecol(BASE / "ecol1\_cleaned.xlsx").rename(columns={"val":"AIR"})

dw = agg\_ecol(BASE / "ecol2\_cleaned.xlsx").rename(columns={"val":"DW"})

msw = agg\_ecol(BASE / "ecol3\_cleaned.xlsx").rename(columns={"val":"MSW"})

# площі (як у попередній комірці)

def normalize\_apostrophe(s): return s.replace("'", "ʼ").replace("’","ʼ").replace("`","ʼ")

raw\_text = """"""

rows = []

for line in [l.strip() for l in raw\_text.strip().splitlines() if l.strip()]:

try:

name, area = line.rsplit(" ", 1)

area = float(area.replace(",", "."))

rows.append((name, area))

except:

pass

areas\_df = pd.DataFrame(rows, columns=["name","area\_km2"])

df\_k = pd.DataFrame({"katottg": codes, "name": names})

df\_k["name\_norm"] = df\_k["name"].apply(normalize\_apostrophe)

areas\_df["name\_norm"] = areas\_df["name"].apply(normalize\_apostrophe)

areas\_final = df\_k.merge(areas\_df.drop(columns=["name"]).rename(columns={"name\_norm":"name\_norm"}),

on="name\_norm", how="left")[["katottg","name","area\_km2"]]

# зібрати грід

grid = areas\_final.merge(tn\_2024, on="katottg", how="left")\

.merge(w\_2024, on="katottg", how="left")\

.merge(air, on="katottg", how="left")\

.merge(dw, on="katottg", how="left")\

.merge(msw, on="katottg", how="left")

for col in ["TN","W","AIR","DW","MSW"]:

grid[col] = grid[col].fillna(0.0)

grid["TN\_km2"] = grid["TN"] / grid["area\_km2"]

grid["W\_km2"] = grid["W"] / grid["area\_km2"]

grid["AIR\_km2"] = grid["AIR"] / grid["area\_km2"]

grid["DW\_km2"] = grid["DW"] / grid["area\_km2"]

grid["MSW\_km2"] = grid["MSW"] / grid["area\_km2"]

# --- локальний Moran з пермутаціями ---

def local\_moran(y, W, nperm=499, seed=42):

rng = np.random.default\_rng(seed)

y = np.asarray(y, dtype=float)

y = np.nan\_to\_num(y, nan=0.0)

z = (y - y.mean())/ (y.std(ddof=1) if y.std(ddof=1)>0 else 1.0)

lag = W @ z

Ii = z \* lag

# пермутації

perm\_I = np.zeros((nperm, len(y)), dtype=float)

for p in range(nperm):

z\_perm = rng.permutation(z)

perm\_I[p,:] = z \* (W @ z\_perm)

# двобічні p-значення поелементно

pvals = (np.sum(np.abs(perm\_I) >= np.abs(Ii), axis=0) + 1) / (nperm + 1)

return Ii, pvals, z, lag

def fdr\_bh(pvals, alpha=0.05):

p = np.asarray(pvals)

n = len(p)

idx = np.argsort(p)

crit = (np.arange(1, n+1) / n) \* alpha

thresh = 0.0

for rank, i in enumerate(idx, start=1):

if p[i] <= crit[rank-1]:

thresh = p[i]

sig = p <= thresh if thresh>0 else np.zeros\_like(p, dtype=bool)

return sig, thresh

vars\_cols = ["TN\_km2","W\_km2","AIR\_km2","DW\_km2","MSW\_km2"]

lisa\_outputs = {}

for col in vars\_cols:

Ii, pvals, z, lag = local\_moran(grid[col].values, W, nperm=999, seed=2025)

sig, thr = fdr\_bh(pvals, alpha=0.05)

quad = np.full(len(z), "NS", dtype=object)

quad[(z>0) & (lag>0) & sig] = "HH"

quad[(z<0) & (lag<0) & sig] = "LL"

quad[(z>0) & (lag<0) & sig] = "HL"

quad[(z<0) & (lag>0) & sig] = "LH"

lisa\_outputs[col] = {"Ii":Ii, "p":pvals, "sig":sig, "thr":thr, "z":z, "lag":lag, "quad":quad}

# --- глобальний Moran для підписів

def moran\_global(y, W, nperm=999, seed=2025):

rng = np.random.default\_rng(seed)

y = np.asarray(y, dtype=float)

y = np.nan\_to\_num(y, nan=0.0)

z = y - y.mean()

z2\_sum = (z\*z).sum()

S0 = W.sum()

I = (len(y)/S0) \* (z @ (W @ z)) / (z2\_sum if z2\_sum>0 else 1.0)

dist = []

for \_ in range(nperm):

z\_perm = rng.permutation(z)

I\_perm = (len(y)/S0) \* (z\_perm @ (W @ z\_perm)) / ((z\_perm\*z\_perm).sum())

dist.append(I\_perm)

dist = np.array(dist)

p = (np.sum(np.abs(dist) >= abs(I)) + 1) / (nperm + 1)

return I, p

# --- функції візуалізації ---

minx = min([s.bbox[0] for s in shapes]); miny = min([s.bbox[1] for s in shapes])

maxx = max([s.bbox[2] for s in shapes]); maxy = max([s.bbox[3] for s in shapes])

padx = (maxx-minx)\*0.03; pady = (maxy-miny)\*0.03

minx -= padx; maxx += padx; miny -= pady; maxy += pady

def map\_lisa(col, out\_png):

out = lisa\_outputs[col]

quad = out["quad"]

# закодуємо категорії числами, щоб не задавати власні кольори (використаємо дефолтну мапу)

cat\_order = ["NS","LL","LH","HL","HH"]

code = np.array([cat\_order.index(q) for q in quad], dtype=float)

# підпис глобального I

Ig, pg = moran\_global(grid[col].values, W, nperm=999)

coll = PatchCollection(patches, linewidths=0.35, edgecolor="black")

coll.set\_array(code) # scalar mappable

# БЕЗ явного задання кольорової схеми — лишаємо дефолт

fig = plt.figure(figsize=(6, 7), dpi=300)

ax = fig.add\_subplot(111)

ax.add\_collection(coll)

ax.set\_xlim(minx, maxx); ax.set\_ylim(miny, maxy)

ax.set\_aspect('equal'); ax.set\_xticks([]); ax.set\_yticks([])

ax.set\_title(f"LISA-карта для {col} (kNN=5, FDR={out['thr']:.3f}); Moran's I={Ig:.3f}, p={pg:.3f}")

cbar = fig.colorbar(coll, ax=ax, fraction=0.03, pad=0.02)

cbar.set\_label("Категорія (0=NS, 1=LL, 2=LH, 3=HL, 4=HH)")

fig.tight\_layout()

fig.savefig(out\_png, bbox\_inches="tight")

plt.close(fig)

def moran\_scatter(col, out\_png):

out = lisa\_outputs[col]

z = out["z"]; lag = out["lag"]

Ig, pg = moran\_global(grid[col].values, W, nperm=999)

fig = plt.figure(figsize=(6, 5), dpi=300)

ax = fig.add\_subplot(111)

ax.scatter(z, lag, s=22, alpha=0.8)

# регресійна лінія через походження з кутовим коефіцієнтом I (для row-std W)

x = np.linspace(z.min(), z.max(), 100)

ax.plot(x, Ig \* x, linewidth=1.5)

ax.axhline(0, linewidth=0.8); ax.axvline(0, linewidth=0.8)

ax.set\_xlabel("z (стандартизоване значення)"); ax.set\_ylabel("Просторовий лаг (W·z)")

ax.set\_title(f"Розсіяння Морана: {col} (I={Ig:.3f}, p={pg:.3f})")

fig.tight\_layout(); fig.savefig(out\_png, bbox\_inches="tight"); plt.close(fig)

# генеруємо

out\_files = []

for col in vars\_cols:

p1 = BASE / f"lisa\_{col}\_knn5.png"

p2 = BASE / f"moran\_scatter\_{col}\_knn5.png"

map\_lisa(col, p1)

moran\_scatter(col, p2)

out\_files += [p1, p2]

# таблиці з підсумком кластерів HH/LL/LH/HL

summary\_rows = []

for col in vars\_cols:

q = lisa\_outputs[col]["quad"]

counts = {lab: int(np.sum(q==lab)) for lab in ["HH","LL","HL","LH","NS"]}

summary\_rows.append({"var": col, \*\*counts})

summary\_df = pd.DataFrame(summary\_rows)

from caas\_jupyter\_tools import display\_dataframe\_to\_user

display\_dataframe\_to\_user("LISA: кількість класифікованих ТГ (kNN=5, FDR 5%)", summary\_df)

# ТОП-5 HH-громад за значенням показника (інтенсивність) для кожної змінної

tops = []

for col in vars\_cols:

q = lisa\_outputs[col]["quad"]

tmp = grid.copy()

tmp["quad"] = q

top = tmp[tmp["quad"]=="HH"].sort\_values(col, ascending=False).head(5)[["name","katottg",col]]

top["var"] = col

tops.append(top)

tops\_df = pd.concat(tops, ignore\_index=True)

display\_dataframe\_to\_user("ТОП-5 HH-кластерів (інтенсивність на км², 2024)", tops\_df)