TP4 Régression multiple et autocorrélation spatiale en Python

2025-2026

1 Les données

Reprendre le jeu de données construit au TP3.

- Réafficher les cartes de chaque variable, et les nuages de points pour chaque paire de variables.

2 Régression multilinéaire

L'objectif de cette partie est d'effectuer une régression linéaire multiple à l'aide de la méthode des moindres carrés.

- Choisissez une variable quantitative à expliquer (ou prédire) en fonction des autres.
- Faites la régression à l'aide de `statsmodel.api.OLS`
- Commentez les métriques données par la méthode `.summary()`
- Quelles sont les valeurs des coefficients ? Certains devraient-ils être nuls ?
- Comparez la carte des prédictions à celle de la variable cible, en utilisant une échelle de couleur commune.
- Faire la carte des résidus
    - Est-ce qu'ils ont l'air visuellement auto-corrélés spatialement
    - Est-ce qu'un individu (c'est-à-dire un pays) a une valeur de résidus extrême ? Si c'est le cas, refaire la régression sans et comparer.

3 Autocorrélation spatiale

Dans cette partie, on va comparer l'autocorrélation spatiale de plusieurs variables.

- Construire la matrice de voisinage W selon les polygones adjacents
- Qui sont les voisins de la France ?
- Les valeurs de la variable à expliquer sont elles similaires pour la France et ses voisins ?
- Faire le diagramme de Moran :
    - Calculer la variable centrée xc = x – x.mean()
    - Obtenir Wtilde en divisant chaque ligne de W par sa somme
    - Remplir les valeurs Nan de Wtilde par des 0 (pas de voisins)
    - Faire la multiplication matricielle `Wtilde @ xc`
    - Tracer le nuage de points
- Calculer les Ii de Moran
    - `Ii = xc * (Wtilde @ xc) / x.var()`

    - Faire la carte des I_i avec une échelle de couleurs pour les pays pour lesquels $I_i > 0$ (pays similaires aux voisins), et une seconde pour ceux pour lesquels $I_i <= 0$ (pays différents des voisins)
    - Calculer I à partir des Ii
- Ecrire une fonction qui calcule le I de Moran pour une variable et l'appliquer aux différentes variables et aux résidus
- **Bonus** : faire la même chose en changeant la définition du voisinage
- **Bonus 2** : faire le même chose avec le C de Geary

4 Régression pondérée géographiquement

- Ecrire une fonction distances(pays_cible) qui renvoie la distance de chaque pays à pays_cible
    - Attention, on ne peut pas juste prendre la distance euclidienne des coordonnées (latitude φ, longitude λ) ! Ni même projeter
    - Approximation sur une sphère de rayon R = 6 371 000m : d = 2*R*arcsin(√a), où $a=\sin^2(\Delta\varphi/2) + \cos(\varphi 1) *(\varphi 2)*\sin^2(\Delta\lambda/2)$, avec les angles exprimés en radians
- Ecrire une fonction w(pays1, pays2, h) qui calcule $w_{i,k}$ pour une valeur de largeur de bande h donnée
- Ecrire une fonction Wi(pays_cible, h) qui fabrique la matrice Wi
- Ecrire une fonction Betas(pays_cible, h) qui calcule $\hat{\beta}(i) = (\tilde{X}^T W_i \tilde{X})^{-1} \tilde{X}^T W_i y$
    - `numpy.linalg.solve(A,B)` permet de calculer $A^{-1}B$ de manière plus stable et rapide que `numpy.linalg.inv(A) @ B`
- Ecrire une fonction predire(pays_cible, h) pour faire l'inférence d'un pays
- Faire la carte des predictions, calculer le R^2 et le RMSE
- Faire les cartes des Betas