

TP2 - Intro à la statistique spatiale 2A - sf

OBJECTIFS DU TP :

- Le but de ce TP est de se familiariser à la manipulation d'objets spatiaux, à l'aide du package **sf**. Vous trouverez un cheatsheet [ici](#)
 - Afin d'utiliser une version de R plus récente (et une version du package **sf** plus récente aussi), vous travaillerez sur le datalab (plateforme du sspcloud, service de l'Insee) : <https://datalab.sspcloud.fr>. Si ce n'est déjà fait, il vous faudra créer un compte utilisateur puis créer un service Rstudio.
 - Si besoin (ce ne sera pas le cas pour cette séance), les fonds disponibles sous "U:/Eleves/Cartographie/Fonds_carte". **PS : Ce répertoire ne doit JAMAIS être votre répertoire de travail !** Il s'agit d'un répertoire où l'on met à disposition des fonds de carte au service de tout le monde. Leur modification pénaliserait donc tous ses utilisateurs.
-

Exercice 1

0. Commencer par créer votre répo github "tp_stats_spatiales". Créer un service Rstudio sur le datalab en saisissant le lien https de votre repo Git (suivre la procédure "Github et le datalab" si besoin). Une fois le service ouvert, créer un projet qui pointe vers votre dossier intitulé "tp_stats_spatiales" (qui est normalement déjà présent à l'ouverture du service).

Charger ensuite les package **sf** et **dplyr**.

1. Importer le fond communal "commune_france_metro_2021.gpkg" disponible dans le dossier "France_metro". Pour cela vous utiliserez la fonction **st_read** du package **sf**. Quelles informations apparaissent dans la console ?
2. Faites un résumé/descriptif du contenu de l'objet importé, comme vous le feriez pour un dataframe.
3. Afficher maintenant les dix premières lignes de la table et regarder la dernière colonne.
4. Afficher le système de projection de la table en utilisant la fonction **st_crs**.
5. Créer une table "communes_Bretagne" ne contenant que les communes bretonnes. Ne conserver que les colonnes (code, libelle, epc, dep, surf) en utilisant la fonction **select()** de **dplyr**. Votre table contient-elle uniquement les 5 variables sélectionnées ?
6. Assurez-vous que cette nouvelle table est toujours un objet **sf**.
7. Appliquer la fonction **plot** sur votre table. (Indice : l'argument **lwd** vous permet de jouer sur l'épaisseur des lignes).
8. Faire la question précédente en utilisant la fonction **st_geometry()** dans votre plot.
9. Créer une variable de surface appelée "surf2" en utilisant les fonctions **st_area()** sur votre variable de geometry. En quelle unité est la variable créée ?
10. Modifier la variable créée pour la convertir en km².
11. Les variables **surf** et **surf2** sont-elles égales ? Pourquoi selon vous ?
12. L'objectif est de créer une table départementale "dept_bretagne" sans doublons. Cette table devra contenir le code département et la superficie du département. Représenter le nouveau fond sur une carte avec la fonction **plot()**.

13. Constituer cette fois un fond départemental en utilisant les fonctions `summarise()` et `st_union()`. À la différence de la table précédemment créée, le fond ne contiendra que le code dept et la geometry (aucune variable numérique ne sera utilisée). Faire ensuite un plot de votre table pour vérifier que les geometry ont bien été regroupés par département.
14. Créer une table “centroid_dept_bretagne” contenant les centroïdes des départements bretons.
 - a. Quel est le type de géometrie associé à ces centroïdes ?
 - b. Représenter les départements bretons et leurs centroïdes sur une même carte, avec deux appels à la fonction `plot()` et en ajoutant l'argument `add = TRUE` sur le second appel.
 - c. Ajouter le nom du département dans le fond de centroïdes. La variable aura pour nom `dept_lib`. Plusieurs solutions sont possibles, la plus propre étant d'utiliser une petite table de passage et de la fusionner avec le fond de centroïdes.
 - d. Récupérer les coordonnées des centroïdes dans un data.frame appelé `centroid_coords` avec la fonction `st_coordinates()`. Observer l'objet obtenu. Ajouter ensuite les colonnes `dep` et `dep_lib` du fond de `centroid_dept_bret` avec la fonction `bind_cols()`. Vous ferez attention à ce que `centroid_coords` ne contienne pas de géométrie (Indice : utiliser `st_drop_geometry()`).
 - e. Représenter les départements, leur centroïde (comme en 14.b) et leur nom (avec la fonction `text()`) sur une seule carte.
15. À l'aide de la fonction `st_intersects()`, retrouver dans quelle commune se situe le centroïde de chaque département breton.
16. Faire la même question avec la fonction `st_intersection()` puis avec la fonction `st_within()`. Quelles différences voyez-vous avec la fonction `st_intersects()` ?
17. Calculer la distance séparant les centroïdes des départements et leur chefs-lieux. Les chefs-lieux des départements bretons sont les communes de Saint-Brieuc (22), Quimper (29), Rennes (35) et Vannes (56). Vous utiliserez la fonction `st_distance()`.
18. Quelles sont les communes à moins de 20 km (à vol d'oiseau) de chaque centroïde ?
 - a. Utilisez la fonction `st_buffer(x, dist)` pour créer une zone de 20 km autour des centroïdes. Vous ferez attention aux unités.
 - b. Représenter sur une carte la géométrie obtenue avec la fonction `plot()`.
 - c. Récupérez les communes comprises dans les tampons obtenus au 18.a avec la fonction `st_intersection()`.
 - d. Combien de communes sont concernées par département ?
19.
 - a. Changer le système de projection des communes bretonnes pour le mettre en WGS84 (EPSG=4326), avec la fonction `st_transform()`.
 - b. Représentez le fond ainsi produit.
20. Recalculer l'aire des communes pour créer une variable `surf3`. Que se passe t'il ?