Ateliers L3 CMI

Quentin Hoarau

2025-09-19

Table of contents

Introduction

Présentation

Ce livre contient les supports de TP des quatre ateliers spécifiques de la L3 CMI de l'université Paris Nanterre :

- Calcul numérique (S2) – 3 ECTS
- SIG : Système d'information géographique (S2) – 3 ECTS
- Econométrie 1 (S1) – 4.5 ECTS
- Econométrie 2 (S2) – 3 ECTS

Notation

Toute la notation est en contrôle continu. Les notes viennent de :

- « Participation » qui inclut l'implication en classe, la qualité des rendus des TP et des pénalités pour retards et absence
- Présentations en classe
- Projets de fin de semestre :
  - Peuvent être communs à plusieurs matières
  - Sujets donnés ou sujets libres
  - Rapport écrit propre (pas de compilation de code, ou de génération IA)
  - Présentation orale avec slides (sans les lire)

Part I

Calcul Numérique

Introduction

- 24 h
- Objectifs : prise en main de R et markdown, statistiques descriptives, graphiques
- Langage : R & Power BI (introduction si a le temps)
- Modalités d'examens :
  - Note de participation
  - Un projet commun calcul numérique/économétrie avec choix du sujet libre
  - Présentation individuelle d'un chapitre d'un livre sur les bonnes pratiques de la dataviz (https://clauswilke.com/dataviz/)

TP1 : Commandes de base de R

1. Manipulation de vecteurs

Soit le vecteur $x = (1, 2, 3, 4, 5)$

1. Créer ce vecteur dans R et le stocker dans un objet que l'on appellera `x` ;
2. Afficher le mode de `x`, puis sa longueur ;
3. Extraire le premier élément, puis le dernier ;
4. Extraire les trois premier éléments et les stocker dans un vecteur que l'on nommera `a`
5. Extraire les éléments en position 1, 3, 5 ; les stocker dans un vecteur que l'on nommera `b`
6. Additionner le nombre 10 au vecteur `x`, puis multiplier le résultat par 2 ;
7. Effectuer l'addition de `a` et `b`, commenter le résultat ;
8. Effectuer l'addition suivante : `x+a`, commenter le résultat, puis regarder le résultat de `a+x` ;
9. Multiplier le vecteur `x` par le scalaire `c` que l'on fixera à 2 ;
10. Effectuer la multiplication de `a` et `b`, commenter le résultat ;
11. Effectuer la multiplication suivante : `x*a`, commenter le résultat ;
12. Récupérer les positions des multiples de 2 du vecteur `x` et les stocker dans un vecteur que l'on nommera `ind`, puis conserver uniquement les multiples de 2 de `x` dans un vecteur que l'on nommera `mult_2` ;
13. Afficher les éléments de `x` qui sont multiples de 3 et multiples de 2 ;
14. Afficher les éléments de `x` qui sont multiples de 3 ou multiples de 2 ;
15. Calculer la somme des éléments de `x` ;
16. Remplacer le premier élément de `x` par un 4 ;
17. Remplacer le premier élément de `x` par la valeur *NA*, puis calculer la somme des éléments de `x`;
18. Lister les objets en mémoire dans la session R ;
19. Supprimer le vecteur `a`;
20. Supprimer la totalité des objets de la session.

2. Manipulation de listes

1. Évaluer le code suivant : TRUE+FALSE+TRUE*4 et le commenter ;

2. Évaluer les expressions suivantes : `c(1, 4, TRUE)`, et `c(1, 4, TRUE, "bonjour")`, commenter ;
3. Créer une liste que l'on appellera `l` et qui contient les éléments 1, 4 et TRUE en première, seconde et troisième positions respectivement ;
4. Extraire le premier élément de la liste `l`, et afficher son mode. En faire de même avec le troisième élément, et commenter ;
5. Ajouter un quatrième élément à la liste `l` : "bonjour", puis afficher la structure de `l` ;
6. Retirer le troisième élément de la liste `l` ;
7. Créer une liste de trois éléments : votre nom, votre prénom, et votre année de naissance. Ces trois éléments de la liste devront être nommés respectivement `"nom"`, `"prenom"` et année de naissance. Stocker la liste ainsi créée dans un objet nommé `moi` ;
8. Extraire le prénom de la liste moi de deux manières : en utilisant l'indice, et en utilisant le nommage ;
9. Créer une liste avec la même structure que celle de `moi`, en la remplissant avec les informations d'une autre personne et la nommer `toi` Puis, créer la liste `personnes`, qui contiendra les listes `toi` et `moi` ;
10. Extraire la liste `toi` de personnes (en première position) ;
11. Extraire directement depuis `personnes` le prénom de l'élément en première position.

Exercice 3 : Manipulation de matrices

1. Créer la matrice suivante :
$$A = \begin{bmatrix} -3 & 5 & 6 \\ -1 & 2 & 2 \\ 1 & -1 & -1 \end{bmatrix}$$
;
2. Afficher la dimension de A, son nombre de colonnes, son nombre de lignes et sa longueur ;
3. Extraire la seconde colonne de A, puis la première ligne ;
4. Extraire l'élément en troisième position à la première ligne ;
5. Extraire la sous-matrice de dimension 2 x 2 du coin inférieur de A
6. Calculer la somme des colonnes puis des lignes de A
7. Afficher la diagonale de A ;
8. Rajouter le vecteur colonne (1, 2 , 3) à droite de la matrice A et stocker le résultat dans un objet appelé B ;

9. Retirer le quatrième vecteur de B ;
10. Retirer la première et la troisième ligne de B ;
11. Ajouter le scalaire 10 à A ;
12. Ajouter le vecteur colonne (1 2 3) à A ;
13. Ajouter la matrice identité I_3 à A ;
14. Diviser tous les éléments de la matrice A par 2 ;
15. Multiplier la matrice A par le vecteur colonne (1 2 3) ;
16. Afficher la transposée de A ;
17. Effectuer le produit avec transposition A^tA.

Exercice 4 : Manipulation de chaînes de caractères

Charger le package `tidyverse`, qui contient le package `stringr`.

1. Créer les objets `a` et `b` afin qu'il contiennent respectivement les chaînes de caractères suivantes : *23 à 0* et *C'est la piquette, Jack!*;
2. Créer le vecteur `phrases` de longueur 2, dont les deux éléments sont a et b ;
3. À l'aide de la fonction appropriée dans le package stringr, afficher le nombre de caractètres de a, de b, puis appliquer la même fonction à l'objet phrases ;
4. En utilisant la fonction `str_c()`, concaténer a et b dans une seule chaîne de caractères, en choisissant la virgule comme caractère de séparation ;
5. Concaténer les deux éléments du vecteur phrases en une seule chaine de caractères, en les séparant par le caractère de retour à la ligne , puis utiliser la fonction `cat()` pour afficher le résultat ;
6. Appliquer la même fonction que dans la question précédente à l'objet suivant : `c(NA, phrases)` et commenter ;
7. Mettre en majuscules, puis en minuscules les chaînes du vecteur phrases (afficher le résultat, ne pas modifier phrases) ;
8. À l'aide de la fonction word() du package stringr, extraire le mot la, puis Jack de la chaîne `b` ;
9. Même question que la précédente, en utilisant la fonction `str_sub()` ;
10. À l'aide de la fonction `str_detect()`, rechercher si le motif `piqu` puis `mauvais` sont présents dans `b` ;
11. À l'aide de la fonction `str_detect()`, rechercher si le motif `piqu` est présent dans les éléments du vecteur phrases ;
12. À l'aide de la fonction `str_detect()`, rechercher si le motif `piqu` ou le motif `à` sont présents dans les éléments du vecteur phrases ;
13. En utilisant la fonction `str_locate()`, retourner les positions de la première occurence du caractère `a` dans la chaîne `b`, puis essayer avec le caractère `w` pour observer le résultat retourné ;

14. Retourner toutes les positions du motif `a` dans la chaîne `b` ;
15. En utilisant la fonction `str_replace()`, remplacer la première occurence du motif `a`, par le motif `Z` (afficher le résultat, ne pas modifier `phrases`) ;
16. Remplacer toutes les occurences de `a` par `Z` dans la chaîne `b` (afficher le résultat, ne pas modifier `phrases`) ;
17. Utiliser la fonction `str_split()` pour séparer la chaîne `b` en utilisant la virgule comme séparateur de sous-chaînes ;
18. Retirer tous les caractères de ponctuation de la chaîne `b`, puis utiliser la fonction `tr_trim()` sur le résultat pour retirer les caractères blancs du début et de la fin de la chaîne.

TP2 : Tableaux de données

On commence par charger les extensions et les données nécessaires.

```
library(tidyverse)
library(nycflights13)
data(flights)
data(airports)
data(airlines)
```

Les verbes de base de `dplyr`

Exercice 1.1

Sélectionner la dixième ligne du tableau des aéroports (`airports`).

Sélectionner les 5 premières lignes de la table `airlines`.

Sélectionner l'aéroport avec l'altitude la plus basse.

Exercice 1.2

Sélectionnez les vols du mois de juillet (variable `month`).

Sélectionnez les vols avec un retard à l'arrivée (variable `arr_delay`) compris entre 5 et 15 minutes.

Sélectionnez les vols des compagnies Delta, United et American (codes `DL`, `UA` et `AA` de la variable `carrier`).

Exercice 1.3

Triez la table `flights` par retard au départ décroissant.

Exercice 1.4

Sélectionnez les colonnes `name`, `lat` et `lon` de la table `airports`

Sélectionnez toutes les colonnes de la table `airports` sauf les colonnes `tz` et `tzone`

Sélectionnez toutes les colonnes de la table `flights` dont les noms se terminent par "delay".

Dans la table `airports`, renommez la colonne `alt` en `altitude` et la colonne `tzone` en `fuseau_horaire`.

Exercice 1.5

Dans la table `airports`, la colonne `alt` contient l'altitude de l'aéroport en pieds. Créer une nouvelle variable `alt_m` contenant l'altitude en mètres (on convertit des pieds en mètres en les divisant par 3.2808). Sélectionner dans la table obtenue uniquement les deux colonnes `alt` et `alt_m`.

Enchaîner des opérations

Exercice 2.1

Réécrire le code de l'exercice précédent en utilisant le *pipe* `%>%`.

Exercice 2.2

En utilisant le *pipe*, sélectionnez les vols à destination de San Francico (code `SFO` de la variable `dest`) et triez-les selon le retard au départ décroissant (variable `dep_delay`).

Exercice 2.3

Sélectionnez les vols des mois de septembre et octobre, conservez les colonnes `dest` et `dep_delay`, créez une nouvelle variable `retard_h` contenant le retard au départ en heures, et conservez uniquement les 5 lignes avec les plus grandes valeurs de `retard_h`.

`group_by` et `summarise`

Exercice 3.1

Affichez le nombre de vols par mois.

Triez la table résultat selon le nombre de vols croissant.

Exercice 3.2

Calculer la distance moyenne des vols selon l'aéroport de départ (variable `origin`).

Exercice 3.3

Calculer le nombre de vols à destination de Los Angeles (code `LAX`) pour chaque mois de l'année.

Exercice 3.4

Calculer le nombre de vols selon le mois et la destination.

Ne conserver, pour chaque mois, que la destination avec le nombre maximal de vols.

Exercice 3.5

Calculer le nombre de vols selon le mois. Ajouter une colonne comportant le pourcentage de vols annuels réalisés par mois.

Exercice 3.6

Calculer, pour chaque aéroport de départ et de destination, la durée moyenne des vols (variable air_time). Pour chaque aéroport de départ, ne conserver que la destination avec la durée moyenne la plus longue.

Jointures

Exercice 4.1

Faire la jointure de la table `airlines` sur la table `flights` à l'aide de `left_join`.

Exercice 4.2

À partir de la table résultat de l'exercice précédent, calculer le retard moyen au départ pour chaque compagnie, et trier selon ce retard décroissant.

Exercice 4.3

Faire la jointure de la table `airports` sur la table `flights` en utilisant comme clé le code de l'aéroport de destination.

À partir de cette table, afficher pour chaque mois le nom de l'aéroport de destination ayant eu le plus petit nombre de vol.

Exercice 4.4

Créer une table indiquant, pour chaque vol, uniquement le nom de l'aéroport de départ et celui de l'aéroport d'arrivée.

Bonus

Exercice 5.1

Calculer le nombre de vols selon l'aéroport de destination, et fusionnez la table `airports` sur le résultat avec `left_join`. Stocker le résultat final dans un objet nommé `flights_dest`.

TP3 : Premiers Graphiques

Exercice 1

1. Avant toute chose, charger `tidyverse`. Charger aussi le jeu de données `rp2018` dans le package `questionr`. Assigner un dataframe `rp69` comme la restriction de `rp2018` aux départements du Rhône et de la Loire. Faire un nuage de points croisant le pourcentage de sans diplôme (`dipl_aucun`) et le pourcentage d'ouvriers (`ouvr`).

2. Faire un nuage de points croisant le pourcentage de sans diplôme et le pourcentage d'ouvriers, avec les points en rouge et de transparence 0.2.

3. Représenter la répartition du pourcentage de propriétaires (variable `proprio`) selon la taille de la commune en classes (variable `pop_cl`) sous forme de boîtes à moustaches.

4. Représenter la répartition du nombre de communes selon la taille de la commune en classes sous la forme d'un diagramme en bâtons.

5. Faire un nuage de points croisant le pourcentage de sans diplôme et le pourcentage d'ouvriers. Faire varier la couleur selon le département (`departement`).

6. Sur le même graphique, faire varier la taille des points selon la population totale de la commune (`pop_tot`).

7. Enfin, toujours sur le même graphique, rendre les points transparents en plaçant leur opacité à 0.5.

9. Représenter la répartition du pourcentage de propriétaires (variable `proprio`) selon la taille de la commune en classes (variable `pop_cl`) sous forme de boîtes à moustaches. Faire varier la couleur de remplissage (attribut `fill`) selon le département.

10. Représenter la répartition du nombre de communes selon la taille de la commune en classes (variable `pop_cl`) sous forme de diagramme en bâtons empilés, avec une couleur différente selon le département.

11. Faire varier la valeur du paramètre `position` pour afficher les barres les unes à côté des autres.

12. Changer à nouveau la valeur du paramètre `position` pour représenter les proportions de communes de chaque département pour chaque catégorie de taille.

13. Faire un nuage de points représentant en abscisse le pourcentage de cadres (`cadres`) et en ordonnée le pourcentage de diplômés du supérieur (`dipl_sup`). Représenter ce nuage par deux graphiques différents selon le département en utilisant `facet_grid`.

14. Sur le même graphique, faire varier la taille des points selon la population totale de la communes (variable `pop_tot`) et rendre les points transparents.

15. Faire le nuage de points croisant pourcentage de chômeurs (`chom`) et pourcentage de sans diplôme. Y ajouter les noms des communes correspondant (variable `commune`), en rouge et en taille 2.5 :

16. Dans le graphique précédent, n'afficher que le nom des communes ayant plus de 15% de chômage.

Exercice 2

Avant tout, charger le package `tidyverse`.

1. Utiliser la fonction `data()` pour charger en mémoire le jeu de données `economics.` Consulter la page d'aide de ce jeu de données pour prendre connaissance de son contenu ;
2. À l'aide de la fonction `ggplot()`, représenter les dépenses personnelles de consommation (`pce`) en fonction de la date (`date`). Les observations doivent être connectées par une ligne.
3. Modifier le graphique de la question précédente de manière à ce que la couleur de la ligne soit dodger blue et définir la taille de la ligne à 0.5. Stocker le résultat dans un objet que l'on appellera `p_1` ;
4. Ajouter une couche au graphique `p_1` pour modifier les titres des axes (les retirer), et définir le titre suivant : *"Personal Consumption Expenditures (billions dollars)"*.
5. Utiliser la fonction `scale_x_date()` du package `scales` pour modifier l'échelle des abscisses de `p_1`, afin que les étiquettes des marques soient affichées tous les 5 ans ;
6. A l'aide de l'option `date_labels()` de la fonction précédente, modifier le format de ces étiquettes pour que seule l'année des dates s'affiche ;

Exercice 3

1. Utiliser la fonction data() pour charger en mémoire le jeu de données `economics.` Consulter la page d'aide de ce jeu de données pour prendre connaissance de son contenu ;

2. Sélectionner les variables `date`,`psavert` et `uempmed` dans le tableau de données economics et utiliser la fonction `gather()` sur le résultat pour obtenir un tableau dans lequel chaque ligne indiquera la valeur (`value`) pour une variable donnée (`key`) à une date donnée (`date`). Stocker le résultat dans un objet que l'on appellera `df` ;

3. Sur un même graphique, représenter à l'aide de lignes, l'évolution dans le temps du taux d'épargne personnelle (psavert) et de la durée médiane en semaines du chômage (`uempmed`). Stocker le graphique dans un objet que l'on appellera `p_2` ;

4. Modifier le code ayant servi à construire le graphique `p_2` pour que le type de ligne soit différent pour chacune des deux séries représentées. Les deux lignes doivent être tracées en bleu. Stocker le graphique dans un objet que l'on appellera `p_3` ;

5. À présent, modifier le code ayant servi à construire `p_3` pour qu'à la fois la couleur et le type de ligne servent à différencier les deux séries. Stocker le graphique dans un objet que l'on appellera `p_4` ;

6. Modifier le graphique `p_4` en ajoutant une couche d'échelle de couleur pour que le taux d'épargne personnelle (`psavert`) soit représenté en dodger blue, et que la durée de chômage (`uempmed`) soit représentée en rouge. Par ailleurs, retirer le titre de la légende ;

7. Modifier le graphique `p_4` en ajoutant une couche d'échelle de type de ligne pour que le taux d'épargne personnelle (`psavert`) soit représenté par des tirets, et que la durée de chômage (`uempmed`) soit représentée par une ligne pleine. Par ailleurs, retirer le titre de la légende des types de lignes, afin que les légendes de couleur et de type de ligne soient fusionnées ;

8. Créer le tableaux de données df_2, une copie de df, dans lequel la variable key doit être un facteur dont les niveaux sont uempmed et psavert ;

9. Créer le vecteur etiq suivant `etiq <- c("psavert" = "Pers. Saving Rate","uempmed" = "Median Duration of Unemployment (weeks)")` Ce vecteur contient des valeurs d'étiquettes pour la légende du graphique qu'il va falloir créer. Représenter sur un même graphique l'évolution dans le temps du taux d'épargne personnelle et de la durée médiane du chômage en semaines, en s'appuyant sur les données contenues dans le tableau `df_2`. La courbe du taux d'épargne personnelle doit être composée de tirets et être de couleur dodger blue; la courbe de la durée médiane du taux de chômage doit être une ligne rouge. La légende ne doit pas comporter de titre, et ses étiquettes doivent être modifiées pour que *"Pers. Saving Rate"* s'affiche 'a la place de *"psavert"*, et pour que *"Median Duration of Unemployment (weeks)"* s'affiche à la place de *"uempmed"*. Stocker le graphique dans un objet que l'on appellera `p_5` ; Note : il s'agit de reprendre le code ayant servi à créer le graphique `p_4`, en modifiant légèrement les échelles de couleur et de ligne pour prendre en compte les étiquettes proposées dans le vecteur `etiq`.

10. Modifier `p_5` pour lui ajouter une couche permettant de déplacer la légende en bas du graphique (utiliser la fonction theme()) ;

11. Ajouter une couche au graphique `p_5` qui permet de définir un thème. Utiliser le thème minimal (`theme_minimal()`). Que se passe-t-il pour la légende ? Repositionner la légende en dessous, et retirer les titres des axes ;
12. Sauvegarder le graphique `p_5` au format PDF en précisant une largeur de 12 et une hauteur de 8.

Exercice 4

1. Charger le jeu de données `mpg` contenu dans lepackage ggplot2 en mémoire, puis consulter la page d'aide du jeu de données pour en prendre connaissance ;
2. Représenter à l'aide d'un nuage de points la relation entre la consommation sur autoroute des véhicules de l'échantillon (`hwy`) et la cylindrée de leur moteur (`displ`)
3. Reprendre le code du graphique précédent et modifier la forme des points pour les changer en symbole +; modifier la couleur des + de manière à la faire dépendre du nombre de cylindres (`cyl`) ;
4. À présent, représenter par des boîtes à moustaches la distribution de la consommation sur autoroute des véhicules (`hwy`) pour chacune des valeurs possibles du nombre de cylindres (`cyl`) ;
5. Charger le jeu de données `economics` contenu dans le package `ggplot2` en mémoire, puis consulter la page d'aide du jeu de données pour en prendre connaissance. Ensuite, ajouter au tableau (les créer) les variables `u_rate` et `e_rate`, respectivement le taux de chômage et le taux d'emploi (on définira le taux de chômage de manière très grossière ici : nombre de personnes non employées sur la population totale) ;
6. Représenter à l'aide de barres l'évolution dans le temps du taux de chômage, et remplir les barres avec la couleur rouge ;
7. Reprendre le code du graphique précédent et ajouter une couche permettant de modifier les limites de l'axe des abscisses pour afficher les valeurs uniquement sur la période "2012-01-01" à "2015-01-01" (utiliser la fonction `coord_cartesian()`). Stocker le graphique dans un objet que l'on appellera p ;
8. Dans le tableau de données economics, sélectionner les variables date, `u_rate` et `e_rate`, puis utiliser la fonction `gather()` pour obtenir un tableau dans lequel chaque ligne correspond à la valeur d'une des variables (taux de chômage ou taux d'emploi) à une date donnée. Stocker le résultat dans un objet que l'on appellera `df_3` ;
9. Utiliser le tableau de données `df_3` pour représenter graphiquement à l'aide de barres les taux de chômage et taux d'emploi par mois sur la période "2012-01-01" à "2015-01-01". Sur le graphique, les barres représentant le taux de chômage et celles représentant le taux d'emploi devront être superposées. Note : il s'agit de modifier légèrement le code ayant permis de réaliser le graphique `p`.

Exercice 5

1. À l'aide de la fonction `WDI` du package `WDI`, récupérer la série fournie par la Banque Mondiale du PIB par tête (*NY.GDP.PCAP.PP.KD*) pour tous les pays disponibles pour l'année 2010, et stocker ces données dans un tableau que l'on appellera `gdp_capita` ;
2. Dans le tableau `gdp_capita`, modifier la valeur de la variable country pour l'observation de la Slovaquie, pour qu'elle vaille Slovakia au lieu de Slovak Republic ;
3. Filtrer les observations du tableau gdp_capita pour ne conserver que les observations des pays membres de l'Union Européenne dont les noms sont contenus dans le vecteur membres_ue. Stocker le résultat dans un tableau que l'on nommera gdp_capita_eu ;
4. Utiliser le package `rworldmap` pour récupérer les données nécessaires à la réalisation d'une carte du monde ;
5. Afficher une carte du monde à l'aide des fonctions contenues dans le package `ggplot2` ;
6. Modifier les échelles des axes pour faire figurer les méridiens de -60 à 60 par pas de 30 et les parallèles de -180 à 180 par pas de 45. Modifier également la projection cartographique pour choisir la projection orthographique, à l'aide de la fonction `coord_map()` ;
7. Joindre les informations contenues dans le tableau `gdp_capita_eu` au tableau contenant les données permettant la réalisation des cartes ;
8. Réaliser une carte choroplèthe reflétant pour chaque pays membre de l'Union Européenne la valeur du PIB par tête de 2012 ;
9. Modifier les couleurs de l'échelle continue de la carte précédente, pour que les faibles valeurs du PIB par tête soient représentées en jaune, et les valeurs les plus hautes en rouge ;
10. Modifier les ruptures de l'échelle de couleur pour qu'elles aillent de 10000 à 100000; modifier également l'étiquette de ces ruptures de sorte que 35000 soit affiché comme 35k, 60000 comme 60k, etc. Enfin, ajouter un titre au graphique et retirer les titres d'axes.

Part II

SIG

Introduction

- 24 h
- objectifs : appréhender les données géographiques et maîtriser les principales opérations sur ce type de données (intersection, distance etc)
- Langage : R & QGIS (introduction)
- modalités d'examens:
  - Note de participation
  - Un projet commun SIG/économétrie avec choix du sujet imposé
  - Présentation en classe d'une carte réalisée avec QGIS : 3mn de présentation

TP1 : introduction à QGIS

1. Prise en main

1.1 Affichage / désaffichage des panneaux

- Fermez les panneaux **Couches** et **Identifier les résultats**.
- Affichez-les de nouveau avec le menu **Vue > Panneaux**.

1.2 Utilisation de données vecteur

1. Examinez la liste des fichiers du répertoire `Data/ADMIN EXPRESS`.
2. Affichez :
   - `ARRONDISSEMENT.shp`
   - `EPCI.shp`
   - `communes_ara.gpkg`
3. Ouvrez la table attributaire de la couche **ARRONDISSEMENT** :
   - Trier selon la colonne `INSEE_DEP`.
   - Quels sont les arrondissements du département de la Loire (42) ?
4. Supprimez la couche **ARRONDISSEMENT**.
5. Ouvrez les propriétés de la couche **COMMUNE** :
   - Notez le système de coordonnées, la géométrie, la liste des attributs.
6. Ouvrez la table attributaire de la couche **COMMUNE** :

- Quel est le nombre de communes ?

7. Identifiez la commune de **Saint-Maurice-en-Gourgois** :
   - Dans quel département est-elle ?
   - Chargez la couche `DEPARTEMENT.shp` et identifiez le département 43.
   - Quelle est sa population (champ `POPULATION`) ?

1.3 Création d'un projet

- Enregistrez le projet sous le nom `TP1-1.qgs`.

1.4 Utilisation de l'outil Identifier les entités

- Utilisez l'outil sur Saint-Maurice-en-Gourgois : quelle est la longueur du périmètre ?
- Zoomez sur la dalle.

1.5 Jointure 1

- Joindre la couche **EPCI** à **COMMUNE**.
- Quel est le nom de l'EPCI de Saint-Maurice-en-Gourgois ?

1.6 Utilisation d'OpenStreetMap

1. Créez une connexion XYZ :

- Nom : `OpenStreetMap`
- URL : `https://tile.openstreetmap.org/{z}/{x}/{y}.png`

2. Installez l'extension **QuickMapServices**.
3. Affichez le fond de carte OSM Standard.

Ordre des couches et opacité

- Classez les couches dans l'ordre : Communes → EPCI → Arrondissement.

Groupe de couches

- Groupez ARRONDISSEMENT et COMMUNE en **ADMIN**.

Outil de mesure

- Mesurez la distance maximale de Saint-Maurice-en-Gourgois.

Sélection et export

- Sélectionnez les communes d'Auvergne-Rhône-Alpes.
- Exportez dans `CommunesEPCI_ARA.gpkg`.
- Créez une couche des seuls EPCI d'Auvergne-Rhône-Alpes.

Sélection et conditions multiples

- Communes AURA $>$ 1000 habitants → combien ?
- Communes Haute-Loire $>$ 1000 habitants → combien ?
- Exportez les deux sélections.

2. Symbologie 1 : Pays du monde

1. Téléchargez `NaturalEarth_TP.gpkg`.
2. Ouvrez :

- `ne_50m_admin_0_countries`
- `ne_50m_populated_places_simple`

- `ne_50m_geographic_lines`

3. Créez une carte par revenu (`INCOME_GRP`).
4. Utilisez palette **Viridis**, projection **World Robinson (EPSG:54030)**.
5. Renommez la couche : *Countries by Income (GDP)*.
6. Enregistrez le projet : `Seance4-1.qgz`.

Mise en page

- Créez une mise en page `gpd` en A4 paysage.
- Ajoutez carte, légende, fond gris clair, export en PNG 300 dpi.

Rendu gradué : Villes du monde

1. Utilisez `pop_max`.
2. Créez 6 classes (Jenks ou seuils manuels).
3. Symbole ponctuel rose, transparence 60 %, taille 0.5–4 mm.
4. Exportez en PNG 300 dpi.

Symboles proportionnels

- Symbole unique, rose, transparence 60 %.
- Taille proportionnelle `pop_max`.
- Enregistrez le projet.

Symbologie 2 : France

- Ouvrir :
- `liste_cheflieu.geojson`
- `COMMUNE.shp`, `REGION.shp`
- `liste-des-gares.geojson`, `RéseauFerré.gpkg`

Styles

- Régions en gris clair + bordures blanches.
- Chefs-lieux : symboles catégorisés (`Préfecture région` et `Préfecture`).
- Réseau ferré : catégorisé sur `type_voie`.
- Communes Occitanie : densité de population (Jenks, OrRd, bornes manuelles).

Étiquettes

- Chefs-lieux avec règles selon statut administratif.

Mise en page finale

- A4 portrait, titre *Réseau ferré en Occitanie*, carte 1/2 000 000, légende, échelle, nord, sources.
- Ajoutez une carte miniature France + Occitanie.
- Export PNG et PDF.

1 TP2 : Traitement sur les données vectorielles

1.1 1. Données de départ

Dans un nouveau projet, ouvrez la couche :
- `DEPARTEMENT_occitanie.gpkg`

Ouvrez également la couche :
- `CLC12_RLRMP_RGF.shp`

Pour visualiser les types d'occupation du sol avec les couleurs standard **Corine Land Cover**, ouvrez les propriétés de la couche et chargez le fichier de style `CLC12.sld` (dans le dossier *FichiersLegende*).

Ajoutez aussi :
- `trace-du-reseau-autoroutier-doccitanie.geojson`
- `dreal-occitanie-mats-eoliens.geojson`

Enregistrez le projet sous le nom `TP2.qgz`.

1.2 2. Création d'un GeoPackage

Toutes les couches produites seront enregistrées dans une base unique **GeoPackage**.

1. Créez `TP2_couches.gpkg` (répertoire *data*) en exportant la couche des mâts éoliens.
2. Options :
   - format = *GeoPackage*
   - nom fichier = `TP2_couches.gpkg`
   - nom couche = `Mâts éoliens`
   - SCR = `EPSG:2154`

1.3 3. Zone tampon sur l'autoroute A61

1.3.1 3.1 Vérification du SCR

Dans les propriétés de `trace-du-reseau-autoroutier-doccitanie`, identifiez le système de coordonnées.

1.3.2 3.2 Conversion en Lambert 93

Exporte la couche vers `TP2_couches.gpkg` avec :
- nom couche = `Réseau autoroutier`
- SCR = `EPSG:2154`

1.3.3 3.3 Sélection et tampon

1. Ouvrir la table attributaire → champ `num_route`.
2. Sélectionner les tronçons correspondant à **A61**.
3. Créer un tampon de **5000 m** avec options :
    - entités sélectionnées uniquement = Oui
    - Nb segments = 10
    - extrémités = Rond
    - regrouper = Oui
    - sortie = `TP2_couches.gpkg` → `Tampon A61 5000m`

1.3.4 3.4 Analyse

Avec **Compter les points dans les polygones**, combien d'éoliennes se trouvent dans cette zone tampon ?

1.4 4. Matrice de distance

Pour chaque mât éolien, calculer la distance avec les **2 plus proches voisins**.
- entrée = `Mâts éoliens`
- identifiant = `id_mat`
- type = *Matrice de distance linéaire (N*k+3)*
- k = 2
- sortie = `TP2_couches.gpkg` → `Calcul Eoliennes 2 voisins`

Inspectez le résultat avec l'outil Identifier : remarquez-vous le type de géométrie ?

1.5 5. Grille hexagonale

1.5.1 5.1 Création

1. Dans `departement_occitanie`, sélectionnez l'Aude et zoomez.
2. Avec **Créer une grille** :
   - type = hexagonale
   - étendue = canevas
   - espacement = 5000
   - SCR = EPSG:2154
   - sortie = `TP2_couches.gpkg` → `Grille Aude 5km`

1.5.2 5.2 Nettoyage

Supprimez les hexagones hors de l'Aude :
- sélection par localisation → inverser → supprimer en mode édition.

1.6 6. Comptages dans la grille

- **Compter points/polygones** → nb d'éoliennes par maille hexagonale.
  - sortie = `TP2_couches.gpkg` → `Calcul nb éoliennes`

- **Somme longueurs lignes** → total autoroutes par maille.
  - sortie = `TP2_couches.gpkg` → `Calcul long autoroutes`

1.7 7. Analyse de superposition (Corine Land Cover)

1.7.1 7.1 Sélection des forêts

Sélectionner dans `CLC12_RLRMP_RGF` les polygones dont `CODE_12` commence par "3".
Exporter vers `TP2_couches.gpkg` → `CLC12 Forets Milieux SemiNat`.

1.7.2 7.2 Superposition

Avec **Analyse de superposition** :
- source = `Grille Aude 5km`
- superposition = `CLC12 Forets Milieux SemiNat`
- sortie = `TP2_couches.gpkg` → `Calcul P ForetsSemiNat`

1.8 8. Intersection et Group Stats

1.8.1 8.1 Zone tampon des parcs

Créer tampon **2000 m** autour de chaque mât → `Tampon Eol 2000m`.
Puis regrouper par `id_parc, n_parc` → `Calcul ParcEol 2000m`.

1.8.2 8.2 Intersection avec CLC

Avec **Intersection** :
- source = `CLC12_RLRMP_RGF`
- superposition = `Tampon Eol 2000m`
- champs conservés : `ID, CODE_12, id_parc, n_parc`
- sortie = `TP2_couches.gpkg` → `Calcul Inter ParcEol CLC12`

1.8.3 8.3 Tableau croisé dynamique

Dans l'extension **Group Stats** :
- Couches = `Calcul Inter ParcEol CLC12`
- Colonnes = `CODE_12`
- Lignes = `id_parc, n_parc`
- Valeurs = `Surface (somme)`

Exporter le tableau et coller dans Excel/Calc.

TP3 : SIG avec R

Chargez les libraries suivantes :

```
#install.packages("spDataLarge", repos = "https://geocompr.r-universe.dev")
#install.packages("remotes")
#remotes::install_github("r-tmap/tmap")

library(tidyverse)
library(sf)
library(stars)
library(terra)
library(spData)
library(spDataLarge)
library(tmap)
library(leaflet)


#remotes::install_github("r-tmap/tmap")
```

1. Premières cartes

1. Décrire l'objet Utilisez `world`. Utiliser `summary()` sur la colonne de géométrie de l'objet `world` inclus dans le package `spData`. Utilisez `ggplot2`. Tracez la carte des continents. Utiliser le `theme_void()`. Tracez le continent asiatique, en filtrant puis appliquant la fonction d'union de formes géométrique `st_union`.

2. Rajouter à la carte des continents des ronds pour chaque pays représentant la racine carré de leur population divisé par 10000. Il faudra pour ça calculer les centroides de chaque pays avec la commande `st_centroid` du package **sf**

3. Tracez la carte de l'Inde dans le continent asiatique. Il faut :

- filtrer et tracer le continent asiatique dans `world` .
- créer un object `india` qui filtre l'Inde dans `world` .
- rajouter la carte le l'Inde en grisant son contour

- créer le centroide de l'inde et rajouter sur la carte une étiquette "Inde" à la coordonnée du centroide du pays

4. Créer un raster de 10x10 pixels avec la commande `rast`, dont les niveaux avec des valeurs aléatoires allant de 0 à 10 (avec la commande `runif`). Tracez ce raster avec `geom_raster`.

5. Chargez le fichier `raster/nlcd.tif` du package `spDataLarge` à l'aide de la commande. Décrire cet objet. Utiliser la fonction `plot`. Enfin, convertir le raster en objet du package `stars` et décrire le résultat.

2. Opérations sur les attributs

Pour ces exercices, nous utiliserons les ensembles de données `us_states` et `us_states_df` du package **spData**.

1. Créez un nouvel objet appelé `us_states_name` qui contient uniquement la colonne `NAME` de l'objet `us_states` en utilisant la syntaxe de base R (`[`) ou tidyverse (`select()`). Quelle est la classe du nouvel objet et qu'est-ce qui le rend géographique?

2. Sélectionnez les colonnes de l'objet `us_states` contenant les données de population. Obtenez le même résultat en utilisant une autre commande (bonus : essayez de trouver trois façons d'obtenir le même résultat). Indice : essayez d'utiliser des fonctions d'aide, telles que `contains` ou `matches` de **dplyr** (voir `?contains`).

3. Trouvez tous les États ayant les caractéristiques suivantes (bonus : trouvez-les *et* affichez-les) :

- Appartiennent à la région Midwest.
- Appartiennent à la région Ouest, ont une superficie inférieure à 250 000 km^2 *et* en 2015, une population supérieure à 5 000 000 d'habitants (astuce : vous devrez peut-être utiliser la fonction `units::set_units()` ou `as.numeric()`).
- Appartiennent à la région Sud, avaient une superficie supérieure à 150 000 km^2 ou une population totale en 2015 supérieure à 7 000 000 d'habitants.

4. Quelle était la population totale en 2015 dans l'ensemble de données `us_states` ? Quelle était la population minimale et maximale en 2015 ?

5. Combien d'États y a-t-il dans chaque région ?

6. Quelle était la population minimale et maximale en 2015 dans chaque région ? Quelle était la population totale en 2015 dans chaque région ?

7. Ajoutez des variables de `us_states_df` à `us_states` et créez un nouvel objet appelé `us_states_stats`. Quelle fonction avez-vous utilisée et pourquoi ? Quelle variable sert de clé dans les deux ensembles de données ? Quelle est la classe du nouvel objet ?

8. `us_states_df` a deux lignes de plus que `us_states`. Comment pouvez-vous les trouver ? (astuce : essayez d'utiliser la fonction `dplyr::anti_join()`)

9. Quelle était la densité de population en 2015 dans chaque État ? Quelle était la densité de population en 2010 dans chaque État ?

10. Combien la densité de population a-t-elle changé entre 2010 et 2015 dans chaque État ? Calculez le changement en pourcentage et cartographiez-le.

11. Changez les noms des colonnes dans `us_states` en minuscules. (Astuce : les fonctions d'aide - `tolower()` et `colnames()` peuvent aider.)

12. Utilisez `us_states` et `us_states_df` pour créer un nouvel objet appelé `us_states_sel`. Le nouvel objet ne doit contenir que deux variables - `median_income_15` et `geometry`. Changez le nom de la colonne `median_income_15` en `Income`.

13. Calculez le changement du nombre de résidents vivant en dessous du seuil de pauvreté entre 2010 et 2015 pour chaque État. (Astuce : voir `?us_states_df` pour la documentation sur les colonnes du seuil de pauvreté.) Bonus : Calculez le changement en pourcentage des résidents vivant en dessous du seuil de pauvreté dans chaque État.

14. Quelle était la population minimale, moyenne et maximale des personnes vivant en dessous du seuil de pauvreté en 2015 pour chaque région ? Bonus : Quelle est la région où l'augmentation du nombre de personnes vivant en dessous du seuil de pauvreté est la plus importante ?

15. Créez un raster `grain` vide avec neuf lignes et colonnes et une résolution de 0,5 degré décimal (WGS84). Remplissez-le avec des nombres aléatoires. Extraire les valeurs des quatre cellules de coin.

16. Quelle est la classe la plus courante de notre exemple de raster `grain` ?

17. Tracez l'histogramme et la boîte à moustaches du fichier `dem.tif` du package **spDataLarge** (`system.file("raster/dem.tif", package = "spDataLarge")`).

3. Opération sur les données spatiales

3.1 Opérations sur les vecteurs

1. Utiliser les jeux de données `nz` et `nz_height` du package `spData`. Combien de ces points élevés la région de Canterbury contient-elle ?

Bonus : tracez le résultat en utilisant la fonction `plot()` pour montrer toute la Nouvelle-Zélande, la région de `Canterbury` en jaune, les points élevés à Canterbury représentés par des croix rouges (astuce : `pch = 7`) et les points élevés dans d'autres parties de la Nouvelle-Zélande représentés par des cercles bleus. Consultez la page d'aide `?points` pour plus de détails avec une illustration des différentes valeurs `pch`.

2. Quelle région a le deuxième plus grand nombre de points `nz_height`, et combien en a-t-elle ?
3. En généralisant la question à toutes les régions : combien des 16 régions de Nouvelle-Zélande contiennent des points qui appartiennent aux 100 points les plus élevés du pays ? Quelles sont ces régions ?

Bonus : créez un tableau listant ces régions par ordre du nombre de points et leur nom.

4. Le point de départ de cet exercice est de créer un objet représentant l'État du Colorado aux États-Unis. Faites ceci avec la fonction `filter()` (`tidyverse`) et tracez l'objet résultant dans le contexte des États-Unis.

- Créez un nouvel objet représentant tous les États qui se chevauchent géographiquement avec le Colorado et tracez le résultat (astuce : la manière la plus concise de le faire est avec la méthode de sous-ensemble `[`).
- Créez un autre objet représentant tous les objets qui touchent (ont une frontière commune avec) le Colorado et tracez le résultat (astuce : souvenez-vous que vous pouvez utiliser l'argument `op = st_intersects` et d'autres relations spatiales lors des opérations de sous-ensemble spatial en R de base).

Bonus : créez une ligne droite allant du centroïde du district de Columbia près de la côte Est au centroïde de la Californie près de la côte Ouest des États-Unis (astuce : les fonctions `st_centroid()`, `st_union()` et `st_cast()` peuvent aider) et identifiez quels États cette longue ligne est.

3.2 Opérations sur les rasters

5. Utilisez `dem = rast(system.file("raster/dem.tif", package = "spDataLarge"))`, et reclassifiez l'élévation en trois classes : basse (<300), moyenne et haute (>500). Ensuite, lisez le raster NDVI (`ndvi = rast(system.file("raster/ndvi.tif", package = "spDataLarge"))`) et calculez la moyenne du NDVI et de l'élévation pour chaque classe d'altitude.
6. Appliquez un filtre de détection de lignes à `rast(system.file("ex/logo.tif", package = "terra"))`. Tracez le résultat. Astuce : Lisez `?terra::focal()`.

7 Calculez l'indice d'eau normalisé (NDWI ; `(green - nir)/(green + nir)`) d'une image Landsat. Utilisez l'image Landsat fournie par le package **spDataLarge** (`system.file("raster/landsat.tif", package = "spDataLarge")`). Calculez également une corrélation entre le NDVI et le NDWI pour cette région (astuce : vous pouvez utiliser la fonction `layerCor()`).

8. Un message sur StackOverflow montre comment calculer les distances jusqu'à la côte la plus proche en utilisant `raster::distance()`. Essayez de faire quelque chose de similaire mais avec `terra::distance()`: récupérez un modèle numérique d'élévation de l'Espagne et calculez un raster qui représente les distances jusqu'à la côte à travers le pays (astuce : utilisez `geodata::elevation_30s()`). Convertissez les distances résultantes de mètres en kilomètres. Note : il peut être judicieux d'augmenter la taille de cellule du raster d'entrée pour réduire le temps de calcul lors de cette opération (`aggregate()`).
9. Essayez de modifier l'approche utilisée dans l'exercice ci-dessus en pondérant le raster de distance avec le raster d'élévation ; chaque tranche de 100 mètres d'altitude devrait augmenter la distance jusqu'à la côte de 10 km. Ensuite, calculez et visualisez la différence entre le raster créé en utilisant la distance euclidienne (E7) et le raster pondéré par l'élévation.

4. Opérations sur les géométries

4.1 Opérations sur les vecteurs

1. Générez et tracez des versions simplifiées de l'ensemble de données `nz`. Expérimentez avec différentes valeurs de `keep` (allant de 0,5 à 0,00005) pour `ms_simplify()` et `dTolerance` (de 100 à 100 000) pour `st_simplify()`.

- À partir de quelle valeur le résultat commence-t-il à se détériorer pour chaque méthode, rendant la Nouvelle-Zélande méconnaissable ?
- Avancé : Quelle est la différence entre le type de géométrie des résultats de `st_simplify()` par rapport au type de géométrie de `ms_simplify()` ? Quels problèmes cela crée-t-il et comment cela peut-il être résolu ?

2. Dans le premier exercice du chapitre sur les opérations de données spatiales, il a été établi que la région de Canterbury avait 70 des 101 points les plus élevés de Nouvelle-Zélande. En utilisant `st_buffer()`, combien de points dans `nz_height` se trouvent à moins de 100 km de Canterbury ?
3. Trouvez le centroïde géographique de la Nouvelle-Zélande. À quelle distance se trouve-t-il du centroïde géographique de Canterbury ?

4. La plupart des cartes du monde ont une orientation nord en haut. Une carte du monde avec une orientation sud en haut pourrait être créée par une réflexion (l'une des transformations affines non mentionnées dans ce chapitre) de la géométrie de l'objet `world`. Écrivez le code pour le faire. Astuce : vous devez utiliser un vecteur à deux éléments pour cette transformation. Bonus : créez une carte à l'envers de votre pays.

5. Exécutez le code de la section 5.2.6. En référence aux objets créés dans cette section, sélectionnez le point dans `p` qui est contenu à la fois dans `x` et `y`.

- Utilisez les opérateurs de sous-ensemble de base.
- Utilisez un objet intermédiaire créé avec `st_intersection()`.

6. Calculez la longueur des lignes de frontières des États-Unis en mètres. Quel État a la frontière la plus longue et lequel a la frontière la plus courte ? Astuce : La fonction `st_length` calcule la longueur d'une géométrie de type `LINESTRING` ou `MULTILINESTRING`. Il faut aussi transformer la géométrieavec un CRS qui puisse calculer des ditances : ici ESPG=2163.

4.2 Opérations sur les rasters

7. Lisez le fichier srtm.tif dans R (`srtm = rast(system.file("raster/srtm.tif", package = "spDataLarge"))`). Ce raster a une résolution de 0,00083 par 0,00083 degrés. Modifiez sa résolution en 0,01 par 0,01 degrés en utilisant toutes les méthodes disponibles dans le package **terra**. Visualisez les résultats. Pouvez-vous remarquer des différences entre les résultats de ces méthodes de rééchantillonnage ?

5. Application : rapprochement de base par distances

Le but de cet exercice est d'identifier la nature de stations de services. Un jeu de données issu de *https://www.data.gouv.fr/fr/datasets/prix-des-carburants-en-france-flux-instantane-v2-amelioree/* donne les prix des carburants mais on n'a pas d'information sur le type de station (station d'autoroute, de supermarché…). Le jeu de données magasins d'openstreetmap pourrait permettre d'apporter des informations.

1. Charger les données de `TP3.zip`. Les gpkg s'ouvrent la commande `st_read` du package `sf`.

2. Restreindre la base `magasins` aux types de magasins (`shop`) suivants : "gas","supermarket","convenience", "car_repair","car","mall","convenience;gas"

3. Transformer les deux jeux en sf dataframe en système de coordonnées EPSG 2154. Attention, pour la base `station`, il faut diviser longitude et latitude par 100000.

4. Pour chaque station station, le magasins le plus proche et calculer la distance correspondate.
5. Quelle est la part des magasins à moins de 100 mètres d'une station.
6. Ajouter les attributs `shop` et `operator` pour chaque magasins les plus proche à la base stations.

Part III

Econométrie 1

Introduction

- 30 h
- objectifs : développer, interpréter et critiquer des modèles économétriques
- modalités d'examens:
- Note de participation
- Un projet commun calcul numérique/économétrie avec choix du sujet libre
- Un projet commun SIG/économétrie avec choix du sujet imposé

TP4 : Probabilités et Statistiques avec R

1. Probabilités avec R

1.1 - Échantillonnage

Vous êtes la fée des loteries dans une loterie hebdomadaire, où 6 numéros uniques sur 49 sont tirés.

1. Tirez aléatoirement les numéros gagnants de cette semaine (fixez la graine à 123) en utilisant la fonction `sample`.

1.2 - Fonction de densité de probabilité

Considérez une variable aléatoire X avec une fonction de densité de probabilité (PDF)

$$f_X(x) = \frac{x}{4} e^{-x^2/8}, \quad x \geq 0.$$

1. Définissez la PDF ci-dessus comme une fonction $f()$.
2. Vérifiez si la fonction que vous avez définie est effectivement une PDF.

1.3 - Espérance et Variance

Dans cet exercice, vous devez calculer l'espérance et la variance de la variable aléatoire X considérée dans l'exercice précédent.

La PDF $f()$ de l'exercice précédent est disponible dans votre environnement de travail.

1. Définissez une fonction appropriée `ex()` qui s'intègre à l'espérance de X.
2. Calculez l'espérance de X. Stockez le résultat dans `expected_value`.
3. Définissez une fonction appropriée `ex2()` qui s'intègre à l'espérance de X^2.
4. Calculez la variance de X. Stockez le résultat dans `variance`.

1.4 - Distribution Normale Standard

Soit $Z\sim \mathcal{N}(0, 1)$.

1. Calculez $\phi(3)$, c'est-à-dire la valeur de la densité de probabilité standard normale en $c=3$.
2. Calculez $P(|Z|\leq 1.64)$ en utilisant la fonction 'pnorm()'.

1.5 - Distribution du Chi-carré

1. Soit $W \sim \chi^2_{1,0}$. Tracez la PDF correspondante à l'aide de `curve()`. Spécifiez la plage de valeurs x comme [0,25] via l'argument `xlim.`
2. Soient X_1 et X_2 deux variables aléatoires normalement distribuées indépendantes avec $\mu = 0$ et $\sigma^2 = 15$. Calculez $P(X_1^2 + X_2^2 > 10)$

1.6 - Distribution de Student

1. Soit $X \sim t_{10000}$ et $Z \sim N(0, 1)$. Calculez le quantile à 95 % des deux distributions. Que remarquez-vous ?
2. Soit $X \sim t_1$. Générez 1000 nombres aléatoires à partir de cette distribution et attribuez-les à la variable `x`. Calculez la moyenne de l'échantillon de x. Pouvez-vous expliquer le résultat ?

1.7 - Distribution de Fisher

1. Soit $Y \sim F(10, 4)$. Tracez la fonction quantile de la distribution donnée à l'aide de la fonction `curve()`.
2. Soit $Y \sim F(4, 5)$. Calculez $P(1 < Y < 10)$ en intégrant la PDF avec la fonction `integrate`.

2. Statistiques avec R

2.1 - Biais

On considère l'estimateur alternatif suivant pour μ_Y, la moyenne de Y :

$$\widetilde{Y} = \frac{1}{n-1} \sum_{i=1}^{n} Y_i$$

1. Définissez une fonction `Y_tilde()` qui implémente l'estimateur ci-dessus.
2. Tirez aléatoirement 5 observations au hasard à partir de la distribution $N(10, 25)$ et calculez une estimation en utilisant `Y_tilde()`. Répétez cette procédure 10000 fois et stockez les résultats dans `est_biased` en utilisant la fonction `replicate`.
3. Tracez un histogramme de `est_biased`. Ajoutez une ligne verticale rouge à $\mu = 10$ en utilisant la fonction `abline()`.
4. Tirez aléatoirement 1000 observations au hasard à partir de la distribution $N(10, 25)$ et calculez une estimation de la moyenne en utilisant`Y_tilde()`. Répétez cette procédure 10000 fois et stockez les résultats dans `est_consistent.`
5. Tracez un histogramme de `est_consistent`. Ajoutez une ligne verticale rouge à $\mu = 10$ en utilisant la fonction `abline()`.

2.2 - Efficience d'un estimateur

Dans cet exercice, nous souhaitons illustrer le résultat selon lequel la moyenne de l'échantillon :

$$\hat{\mu}_Y = \sum_{i=1}^{n} a_i Y_i$$

avec le schéma de pondération égale $a_i = \frac{1}{n}$ pour $i = 1, ..., n$ est l'estimateur linéaire non biaisé meilleur (BLUE) de μ_Y.

En tant qu'alternative, considérez l'estimateur :

$$\tilde{\mu}_Y = \sum_{i=1}^{n} b_i Y_i$$

où b_i donne aux premières $\frac{n}{2}$ observations un poids plus élevé de 3 que les deuxièmes $\frac{n}{2}$ observations (nous supposons que n est pair pour simplifier).

%Le vecteur de poids w a déjà été défini et est disponible dans votre environnement de travail.

1. Définissez un vecteur de pondération pour une taille d'échantillon `n=100`. Il doit être normalisé.
2. Vérifiez que $\tilde{\mu}_Y$ est un estimateur non biaisé de μ_Y, la moyenne de Y_i.
3. Implémentez l'estimateur alternatif de μ_Y en tant que fonction `mu_tilde()`.

4. Tirez au hasard 100 observations à partir de la distribution $\mathcal{N}(5, 10)$ et calculez les estimations avec les deux estimateurs. Répétez cette procédure 10000 fois et stockez les résultats dans `est_bar` et `est_tilde`. Utilisez la fonction `replicate`.
5. Calculez les variances de l'échantillon de `est_bar` et `est_tilde`. Que pouvez-vous dire sur les deux estimateurs?

2.3 - Test d'hypothèse

Considérez l'ensemble de données `wage1` du package `wooldridge`. La variable `wage` donne les gains horaires moyens des individus. Nous supposons que les gains horaires moyens `wage` dépassent 10 dollars par heure et souhaitons tester cette hypothèse à un niveau de signification de $\alpha = 0,05$. Veuillez faire ce qui suit :

1. Calculez la statistique de test manuellement et attribuez-la à `tstat`.
2. Utilisez `tstat` pour accepter ou rejeter l'hypothèse nulle.
3. Refaites-le en utilisant l'approximation normale.
4. Calculez la valeur-p manuellement et attribuez-la à `pval` en utilisant l'approximation normale.
5. Utilisez `pval` pour accepter ou rejeter l'hypothèse nulle.
6. Effectuez le test d'hypothèse des questions précédentes en utilisant la fonction `t.test()`.
7. Extrayez la statistique t et la valeur-p de la liste créée par `t.test()`. Attribuez-les aux variables `tstat` et `pvalue`.
8. Vérifiez que l'utilisation de l'approximation normale ici est également valide en calculant la différence entre les deux valeurs-p.

2.4 - Test d'hypothèse : valeur-p

On considère les données CO2 (`data(CO2)`).

1. Tester s'il existe une différence significative dans l'absorption entre les plantes traitées et les plantes non traitées à un niveau de signification de α=0,05.
2. Obtenez l'intervalle de confiance.

2.5 - Corrélation

Charger la librarie `corrgram` et le jeu de données `auto`.

1. Calculez la corrélation simple (linéaire) entre le prix de la voiture (`Price`) et son économie de carburant `MPG` (mesurée en miles par gallon, ou mpg).
2. Utilisez la fonction `cor.test` pour vérifier si le coefficient obtenu est statistiquement significatif au niveau de 5 %.
3. La corrélation simple suppose une relation linéaire entre les variables, mais il peut être utile de relâcher cette hypothèse. Calculez le coefficient de corrélation de Spearman pour les mêmes variables et trouvez sa signification statistique.
4. En R, il est possible de calculer la corrélation pour toutes les paires de variables numériques dans un dataframe en une seule fois. Cependant, cela nécessite d'exclure d'abord les variables non numériques. Créez un nouveau dataframe, `auto_num`, qui ne contient que les colonnes avec des valeurs numériques du dataframe `auto`. Vous pouvez le faire en utilisant la fonction `filter`.
5. Utilisez la fonction `cor` pour créer une matrice de coefficients de corrélation pour les variables du dataframe `auto_num`.
6. La fonction standard `cor.test` ne fonctionne pas avec des dataframes. Cependant, la signification statistique des coefficients de corrélation pour un dataframe peut être vérifiée à l'aide de la fonction `rcorr` du package `Hmisc`. Transformez le dataframe `auto_num` en une matrice (`auto_mat`) et utilisez-le pour vérifier la signification des coefficients de corrélation avec la fonction `rcorr`.
7. Utilisez la fonction `corrgram` du package `corrgram` pour créer un correlogramme par défaut afin de visualiser les corrélations entre les variables du dataframe `auto`.
8. Créez un autre correlogramme qui (1) ne comprend que le panneau inférieur, (2) utilise des diagrammes en camembert pour représenter les coefficients de corrélation et (3) ordonne les variables selon l'ordre par défaut.
9. Créez un nouveau dataframe, `auto_subset`, en sous-échantillonnant le dataframe `auto` pour inclure uniquement les variables `Price`, `MPG`, `Hroom` et `Rseat`. Utilisez le nouveau dataframe pour créer un correlogramme qui (1) affiche les coefficients de corrélation dans le panneau inférieur et (2) montre des diagrammes de dispersion (points) dans le panneau supérieur.
10. Utilisez la fonction `correlations` du package `ggm` pour créer une matrice de corrélation avec à la fois des coefficients de corrélation complets et partiels pour le dataframe `auto_subset`. Trouvez la corrélation partielle entre le prix de la voiture et son économie de carburant.

Projet 1 - Analyse des Disparités Scolaires : Impact des Facteurs Socio-Économiques sur les Résultats du Brevet des Collèges

Les inégalités de performance scolaire sont un sujet récurrent dans les débats sur le système éducatif. Parmi les examens importants en France, le brevet des collèges permet de mesurer les compétences acquises par les élèves à la fin du cycle secondaire. Cependant, les résultats obtenus peuvent varier en fonction de divers facteurs, notamment le contexte socio-économique local.

Ce TP vous propose d'explorer l'influence de facteurs socio-économiques, tels que le revenu médian, le taux de chômage ou encore le niveau d'éducation dans les communes, sur les résultats du brevet des collèges. À travers l'analyse de jeux de données réels, vous serez amenés à identifier des corrélations et à mieux comprendre les déterminants de la performance scolaire.

0. Installation.

Charger les packages `tidyverse`, `stargazer`. ChatGPT ou autre chatbot sont autorisés pour ce TP.

Les données socio-économiques sont bien formatées ici : https://www.unehistoireduconflitpolitique.fr/telecharger.html. Commencer par télécharger les données sur les revenus des communes. On pourra réitérer l'analyse sur les diplômes et les catégories socio-professionnelles.

1. Chercher sur internet et télécharger les données sur les résultats de brevets par établissement.

1. Données Brevet

1.1 Description des données

1. Décrire le jeu de données : colonnes, taille, niveau géographique, horizon temporel...
2. Quelle est la période étudiée ?

3. Combien y a-t-il d'établissements ?

1.2 Evolution temporelle

On veut caractériser les résultas du brevet au niveau national.

1. Créer un fichier de donnée aggrégé par année au niveau national (utiliser `group_by` et `summarize`).
2. Comment semble calculé la colonne `taux_de_reussite`. Tester son intuition une colonne `taux_de_reussite2` et comparer avec `taux_de_reussite`.
3. Faire des graphiques montrant l'évolution des nombres d'inscrits et d'admis.
4. Faire des graphiques montrant le taux annuel d'admis.
5. Faire des graphiques montrant les taux annuels d'admis pour chaque mention.
6. Décrire et interpréter chaque graphiques.

1.3 Variation en coupe

On considère la dernière session reportée par le jeu de donnée.

1. Créer un jeu de donnée filtré sur cette dernière année.
2. Montrer es graphiques en barres pour représenter les différences sur les taux de réussites selon le type d'établissement et le secteur d'enseignement.
3. Faire des classements des dix meilleurs départements selon les différents taux de réussites.

2. Données socio-économiques

2.1 Description du jeu de données

1. Décrire le jeu de données de la même façon que pour le premier jeu. Utiliser les annexes où les données sont décrites.
2. Quelles colonnes (ou ensemble de colonnes) vous semble-t-il pertinent de garder ?

2.2 Transformation du jeu

Transformer ce jeu de données en format "long" avec `pivot_longer`.

3. Analyse jointe .

3.1 Jointure

Pour chaque année et pour chaque établissement, on souhaite avoir les informations socio-économiques de la commune correspondante.

1. Faire la jointure entre les deux jeux de données.
2. Analyser les données manquantes du nouveau jeu de données.

3.2 Analyse en coupe

1. Faire des graphiques par points représentant le revenu moyen de la commune de l'établissement avec ses différents taux de réussites.
2. Faire des graphiques en bar dans lequel par décile de revenu (utiliser la colonne de percentile coté socioeco).

3.3 Regressions linéaire

Pour une année donnée vs toutes taux de reussite en fonction de la taille de la commune, revenus moyen

1. Faites une régression

Notes: - on fait les régressions avec la commande `lm`. - Pour visualiser les régressions, on enregistre les résultats de chaque regressions (eg, `lm1`,`lm2`…) et on visualise avec la commande `stargazer` du package du même nom (eg `stargazer(type="text",lm1,lm2)`).

Projet 2 - Etude économétrique de l'Enquête Nationale Transport 2019

1. Enoncé

Dans ce TP, nous allons travaillons sur enquête nationale de l'INSEE. Vous aurez la liberté de choisir une question de recherche et de sélectionner les variables qui vous semblent pertinentes (en n'en prenant pas trop tout de même).

Les données sont disponibles ici : **https://www.statistiques.developpement-durable.gouv.fr/resultats-detailles-de-lenquete-mobilite-des-personnes-de-2019**.

On considère les caractéristiques socio-économiques des ménages suivantes : le revenu, la catégorie socio-professionnelle, le lieux de résidence, la composition du ménage (nombre de personnes, age).

Questions : comment varie les grandeurs suivantes en fonction des grandeurs suivantes:

- les caractéristiques du véhicule : age, motorisation
- distance et nombre de trajets parcourue à velo pour les trajets du quotidient
- distance et nombre de trajets parcourue en voiture en commun pour les trajets du quotidient
- distance et nombre de trajets parcourue en transport en commun pour les trajets du quotidient
- distance et nombre de trajets parcourue en avion pour les voyages

Travail à faire :

1. identifier dans les jeux de données où trouver les informations pertinentes
2. effectuer des statistiques descriptives sur les caractéristiques socio-économiques
3. construire votre jeu de donnée en construisant les grandeurs de transport puis en réalisant en appariemment sur les données socioéconomiques.
4. Effectuer des régressions linéaires en combinant différemment les variables de controlms.

Part IV

Econométrie 2

Introduction

- 24 h
- objectifs : économétrie avancée : test (en parallèle du cours) et introduction à l'inférence causale
- modalités d'examens :
  - Note de participation (30%)
  - Présentations en groupe en classe de chapitre d'un manuel d'inférence causale (30%)
  - Projet libre d'économétrie