Université de Nantes

BUT1 Info

Statistiques descriptives

2023-2024

Exercice 1 Comptoir de l'espace

Vous disposez de deux fichiers `csv` pour mener une étude.

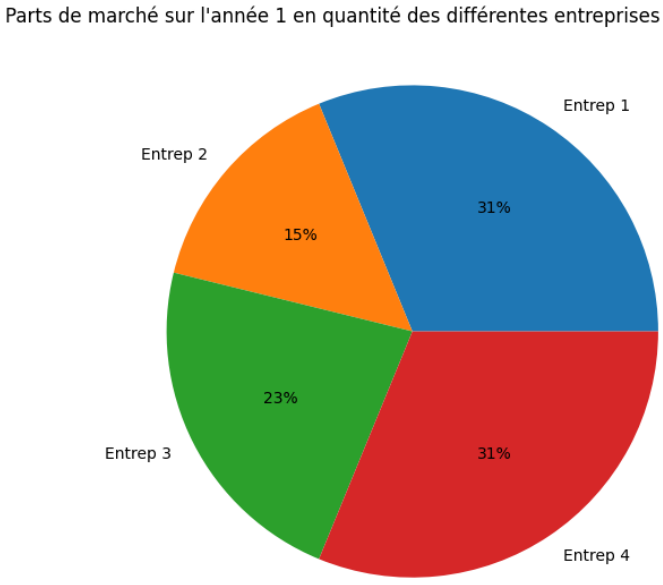
Un premier fichier donne le positionnement de 5 entreprises vendant des places dans un comptoir de l'espace pour des voyages. Ces entreprises sont en concurrence sur deux types de voyages potentiels désignés par un numéro de produit. Si elles se positionnent sur un certain produit elles décident du prix du billet, de la qualité du confort à bord (via un coût consacré) et d'un budget de communication pour ce produit.

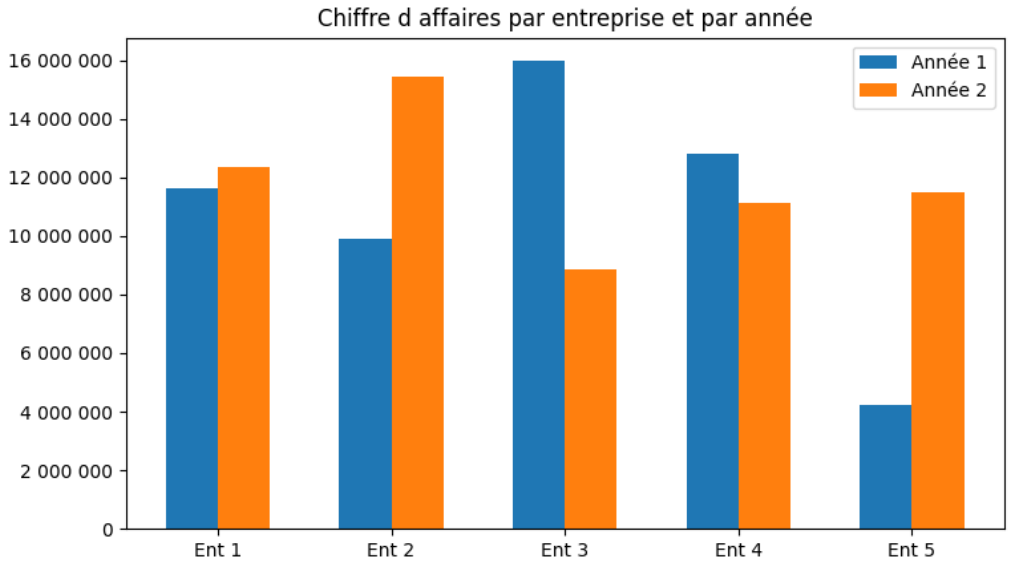
Un deuxième fichier donne les attentes des clients se rendant au comptoir de l'espace. Ils sont interrogés pour :

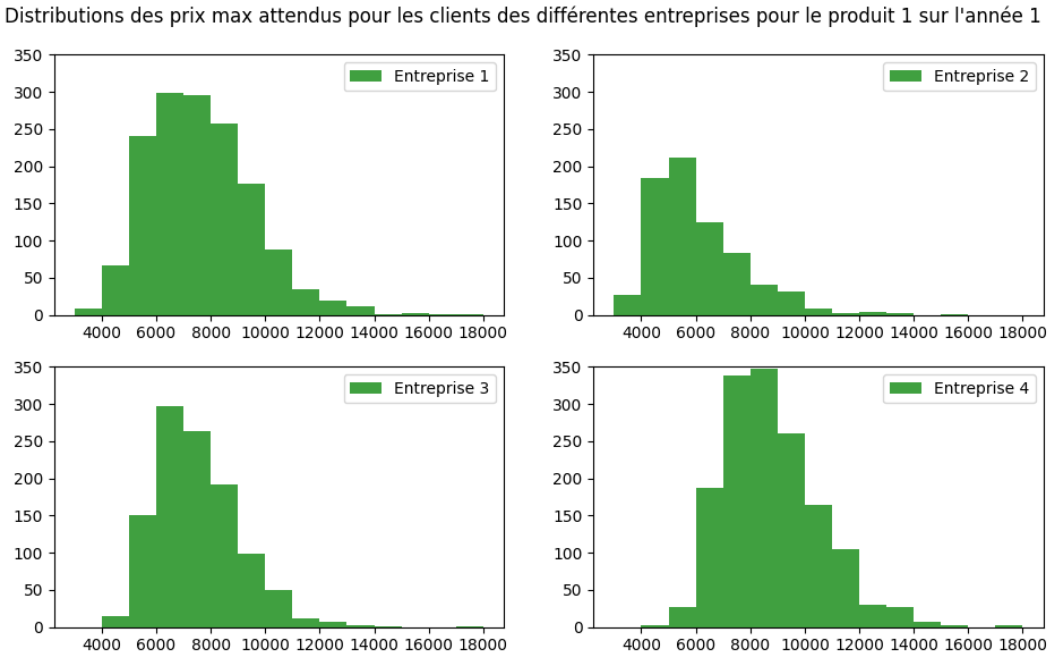
- connaître le produit qui les intéresse
- estimer le prix max qu'ils seraient prêt à consacrer à l'achat d'un billet a priori et le coût en confort min que les entreprises devraient consacrer pour les satisfaire
- évaluer l'importance que semble avoir pour eux la communication (évalué sur une échelle de 1 à 5)
- prendre quelques renseignements sur eux

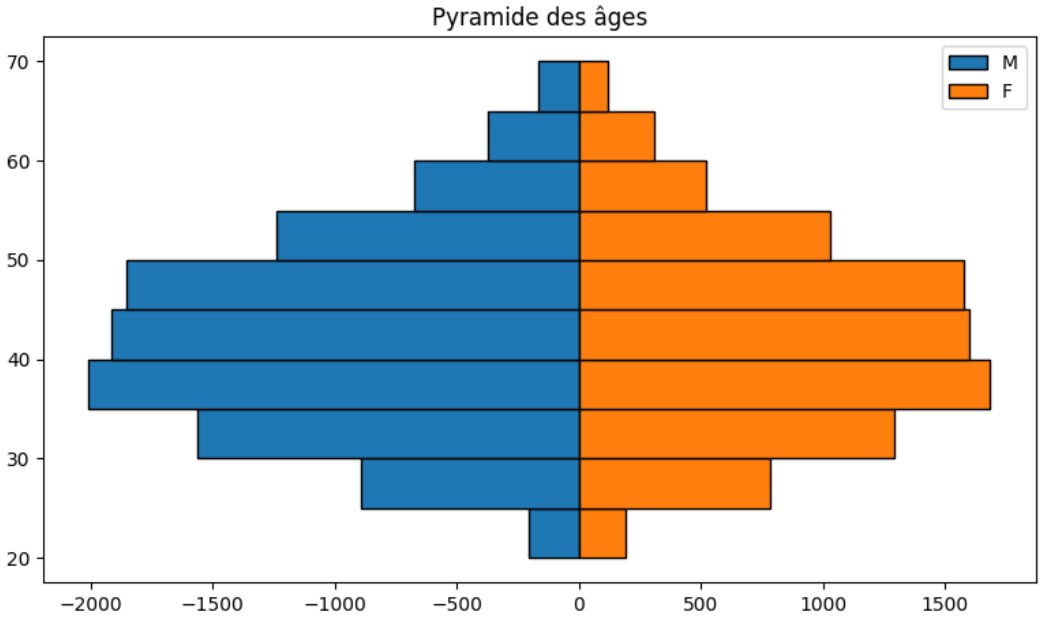
A la sortie ils déclarent s'ils ont acheté un billet pour une entreprise ou non (cas `CLI_CHOIX=0`)

1.1. Produire les graphiques joints à partir de données extraites de ces deux tables et donner pour chacun de ces graphiques une interprétation de la représentation en choisissant un cas particulier (i.e. : Donner par exemple sur le premier graphique une phrase précisant l'information contenue dans la part représentée en bleu)



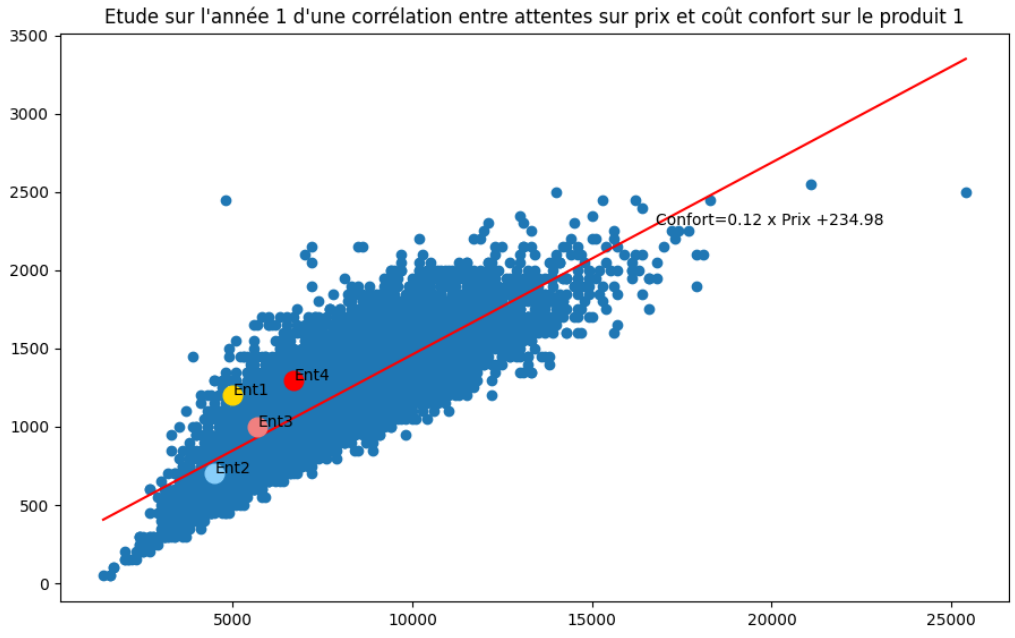

Chiffre d affaires par entreprise et par année
Année 1
Année 2
16 000 000
14 000 000
12 000 000
10 000 000
8 000 000
6 000 000
4 000 000
2 000 000
0
Ent 1
Ent 2
Ent 3
Ent 4
Ent 5

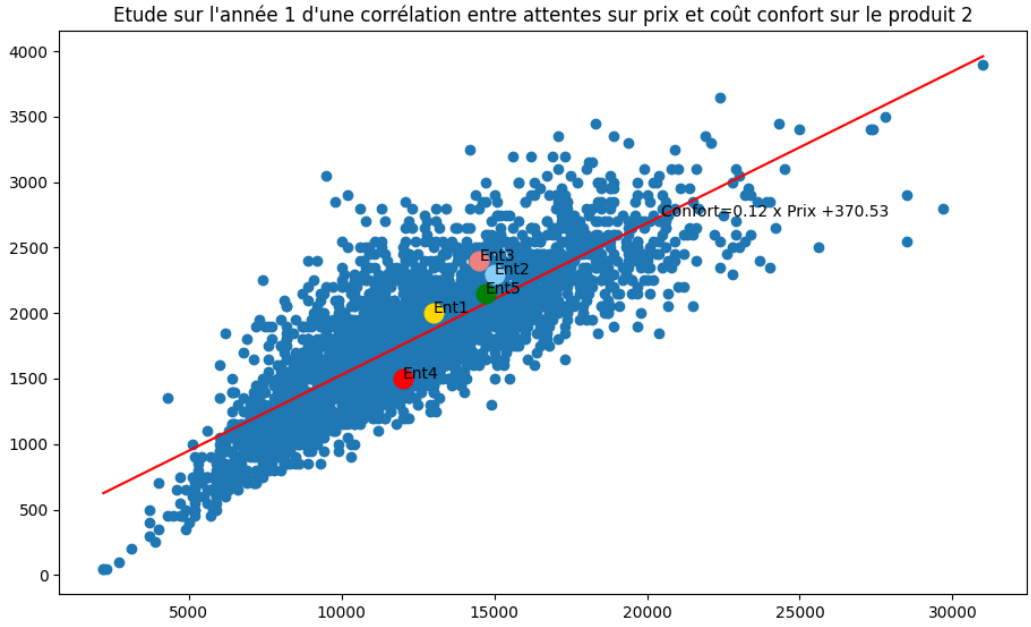

Distributions des prix max attendus pour les clients des différentes entreprises pour le produit 1 sur l'année 1
Entreprise 1
Entreprise 2
Entreprise 3
Entreprise 4
350
300
250
200
150
100
50
0
4000 6000 8000 10000 12000 14000 16000 18000



Pour les graphiques qui suivent on utilisera la représentation en nuage de points `scatter` de `matplotlib` et la fonction `linregress` de la bibliothèque `scipy` pour interroger les résultats obtenus pour un modèle de régression linéaire.

```
from scipy.stats import linregress
res_reg=linregress(var_explicative,var_expliquee)
a=res_reg.slope
b=res_reg.intercept
coef_cor=res_reg.rvalue
```





1.2. D'après le modèle défini précédemment, que peut-on prévoir comme coût de confort minimal sur le produit 1 pour un client ayant un prix maximal envisagé de 14 000 €.

1.3. Commentez le positionnement des entreprises par rapport aux liens entre prix et confort attendus définis par les modèles représentés.

1.4. En dehors de considérations liées au prix mais en considérant le fait qu'il y ait naturellement un coût confort attendu sur le produit 2 supérieur à celui attendu sur le produit 1 on veut dresser un classement sur l'ensemble de la clientèle quant à l'importance accordée au confort. Que peut-on proposer ?

Exercice 2 On va taper la balle ?

Vous disposez de différents fichiers `csv` relatifs à des données tennistiques (ATP) sur la période de 1991 à 2017. Vous allez devoir dans un premier temps importer le contenu de ces fichiers en créant différents `dataframe` qu'on liste ci-dessous :

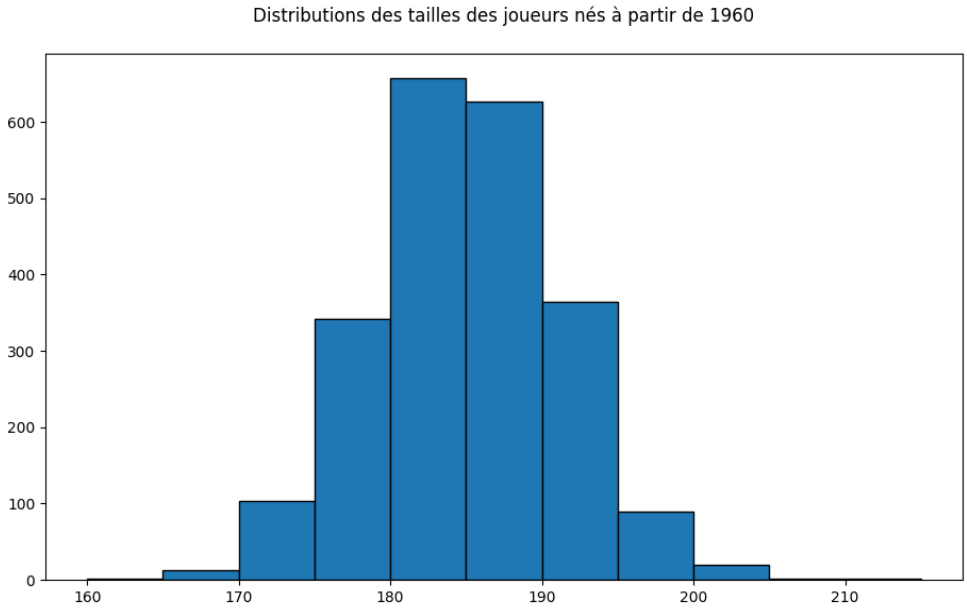
- `tournois`
- `joueurs`
- `match_scores`
- `match_stats`
- `classement` (à créer à partir de plusieurs fichiers de données)

2.1. On souhaite étudier les tailles de joueurs pour l'ensemble des joueurs nés au plus tôt en 1960.

a. Pour cela il nous faut commencer par nettoyer un peu les données en ne conservant que les données des joueurs plus grands qu'une balle de tennis (de diamètre 6.5 cm) et en veillant toujours à la date de naissance (qu'on paramétrera par une variable de sorte à ce qu'elle puisse être modifiée). Récupérer un dataframe pour la taille des joueurs obéissant à ces critères.

b. Afficher le nombre de joueurs concernés, la taille moyenne, la taille médiane et l'écart type des tailles.

c. Réaliser un histogramme montrant la répartition de ces tailles en imposant le choix des tranches avec des bornes définies comme des multiples de 5 et de sorte à ce que le calcul de la première borne inférieure et la dernière borne supérieure soit effectué automatiquement en fonction des valeurs à représenter. Vous devriez obtenir le graphique ci-dessous :



2.2. On veut présenter un graphique montrant l'évolution du classement d'un joueur dont on aura saisi et enregistré l'identifiant dans une variable.

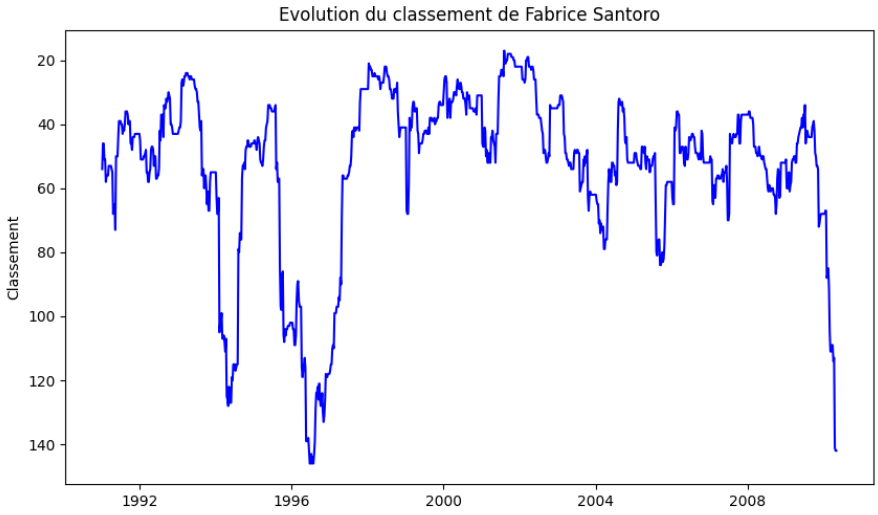
a. Il nous faut ainsi récupérer les dates pour lesquelles le joueur a bien eu un classement associé et formater celles-ci pour qu'elles puissent être représentées

en abscisses du graphique voulu. Pour cela on pourra réordonner le dataframe lié aux classements par ordre croissant sur les enregistrements du champ `week_title` avec la méthodes `sort_values` (vous pouvez vérifier que l'ordre lexicographique utilisé est compatible avec un ordre chronologique avec le format utilisé ici pour les dates) et :

- soit importer le module `datetime` à exploiter pour effectuer la transformation des données de la series issue de `week_title` en un tableau numpy exploitable graphiquement avec la précision du jour, du mois et de l'année.
- soit transformer le tableau numpy de valeurs issues de la series `week_title` de sorte à ce qu'il puisse être converti en un tableau numpy d'objets de type `datetime64`.

Remarque : A défaut d'avoir réordonné le dataframe, la méthode `numpy.argsort` pourrait également être utilisée pour obtenir un index permettant de s'assurer de la croissance des dates.

b. Réaliser le graphique présenté ci-dessous pour le joueur d'identifiant `s424` (on aura pris soin d'inverser l'ordre de représentation des ordonnées avec la méthode `invert_yaxis()` pour une lecture plus naturelle dans ce contexte).



2.3. On veut réaliser une étude quant à une corrélation potentielle entre la taille d'un joueur et le taux de points gagnés suite à un service sur première balle réussi.

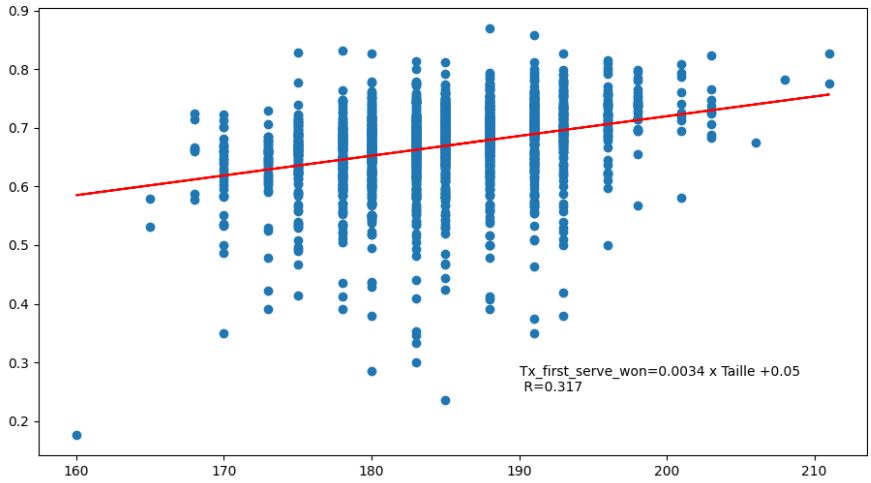
a. Pour cela il faut de nouveau nettoyer un peu la base pour ne garder que les données de `match_stats` pour lesquelles les nombres de points gagnés sur premiers services (`winner_service_points_won` et `loser_service_points_won`) sont bien bien inférieurs aux nombres de premiers services réussis (`winner_first_serves_in` et `loser_first_serves_in`) et que le nombre de points gagnés sur premier service est strictement supérieur à 0 et ceci à la fois pour les gagnants d'un match et les perdants.

On veut pouvoir calculer pour chaque joueur le taux de points gagnés après un premier service réussi.

Pour cela il est conseillé de chercher à extraire un dataframe `stats_winner` donnant pour chaque joueur le cumul sur ses matchs gagnés des informations intéressantes sur ses services et d'autre part un autre dataframe `stats_loser` donnant à nouveau pour chaque joueur le cumul sur ses matchs perdus des informations intéressantes sur ses services.

In fine on pourra rassembler les deux extractions dans un seul dataframe par concaténation sur les colonnes. Dans ce dernier dataframe on pourra rajouter une colonne donnant le taux de réussite attendu pour représenter enfin son lien avec la taille du joueur.

b. Réaliser le graphique présenté ci-dessous



Exercice 3 Communes de France

Vous disposez de deux fichiers :

- `RGC_2013.csv` vous donnant des informations générales (Population, surface...). L'identification de chacune des communes s'y effectuant à partir d'un identifiant de département `DEP` et de l'identifiant de la commune à l'intérieur de son département `COM`.
- `Emplois_salaries` donnant les effectifs salariés dans différents secteurs d'activités par commune ou arrondissement municipal (pour Lyon, Marseille et Paris). Chaque ligne est identifiée par le code géographique `CODGEO` obtenu en concaténant les chaînes de caractères désignant le numéro de département et le numéro d'une commune dans son département.

3.1. Pour exploiter les liens possibles entre les deux tables, il nous faut commencer par traiter un peu les données des deux dataframes extraits (`RGC` et `EMPLOIS_SALARIES`) de sorte à créer une association correcte entre les deux tables.

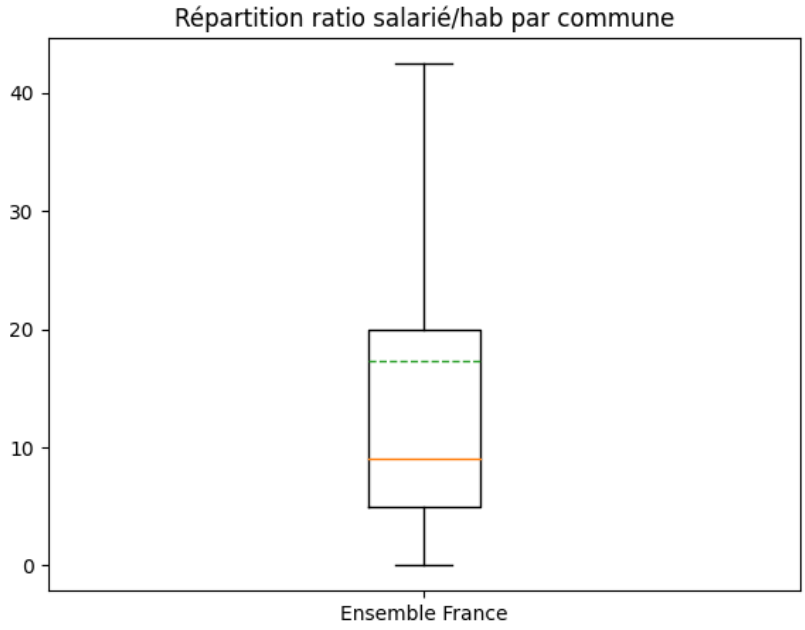
a. Transformer le type de données manipulées dans les colonnes `DEP`,`COM` du dataframe `RGC` et sur la colonne `CODGEO` du dataframe `EMPLOIS_SALARIES`, de sorte à ce que celui-ci soit globalement défini comme `str`. On pourra pour cela exploiter la méthode `astype(type)` disponible pour un dataframe.

b. Insérer en troisième colonne du dataframe RGC (méthode `insert(loc,column,value` pour un dataframe) une colonne nommée `COG` et réalisant la concaténation voulue (on pourra s'aider de la méthode `str.zfill(int)`)

Les deux dataframes doivent maintenant pouvoir être associés à partir de leurs identifiants `COG` et `CODGEO`. Les communes de Lyon(`COG=69123`, Marseille (`COG=13055`) et Paris (`COG=75056`) ayant leurs emplois salariés uniquement enregistrés selon le détail de ceux-ci par arrondissement municipal (`CODGEO` 69381 à 69389 pour Lyon, 13201 à 13216 pour Marseille et 75101 à 75120 pour Paris), vous devrez néanmoins rajouter une ligne pour chaque commune dans le dataframe `EMPLOIS_SALARIES` produisant la somme des effectifs pour les arrondissements correspondant (et avec un `CODGEO` identique au `COG` de la commune).

c. Créer une fonction `associate_arr_mun(num_dep,cog,num_arr_debut,num_arr_fin)` effectuant cet ajout puis l'appliquer aux 3 cas (par exemple `associate_arr_mun(69,69123,381,389)`). La méthode `isin(list_values)` applicable à une `series` permet de renvoyer un mask ce qui peut-être intéressant ici)

3.2. On s'intéresse au ratio de l'effectif salarié par habitant sur l'ensemble des communes (on supprimera pour cette seule question les communes aux populations nulles et les communes pour lesquelles le ratio calculé renvoie la valeur `nan`). Réaliser le graphique ci-dessous (`plt.boxplot`). Pouvez-vous expliquer la différence significative entre la moyenne et la médiane ?



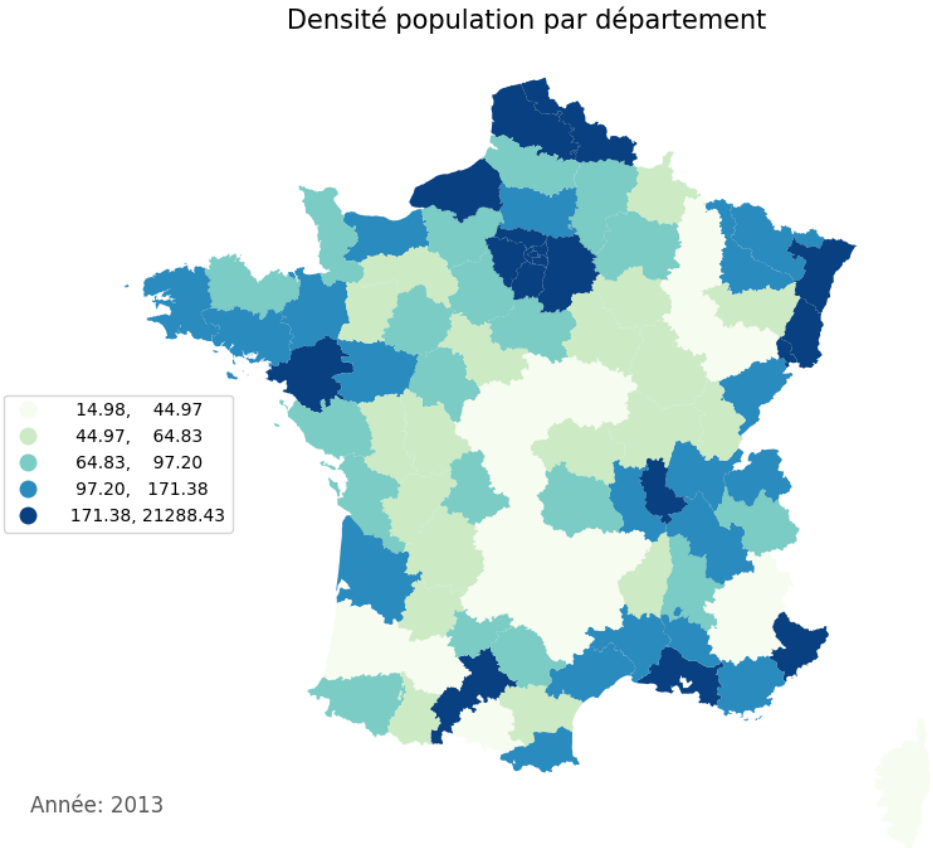
3.3. On s'intéresse cette fois-ci au ratio de l'effectif salarié par habitant sur l'ensemble d'un département. Représenter le classement des dix départements ayant les ratios les plus élevés.

Pour les questions suivantes on veut pouvoir visualiser géographiquement des données extraites de nos dataframes. Pour cela vous allez devoir installer deux packages en rentrant les commandes suivantes dans le terminal `pip install geopandas` et `pip install mapclassify`. On va par ailleurs s'appuyer sur deux fichiers représentant d'une part les limites des départements français `departements.geojson` et d'autre part les limites des pays `countries.geojson`. On peut dès lors créer des `geodataframe` auxquels on pourra appliquer la méthode plot qui détectera la column geometry pour représenter les contours des zones géographiques. Par exemple :

```
import geopandas as gpd
world = gpd.read_file("countries.geojson")
fig, ax = plt.subplots(1, figsize=(10,10))
world_map = gpd.GeoDataFrame(world)
world_map.plot(ax=ax)
#Et si on veut restreindre la zone à la France métropolitaine :
'''ax.set_xlim(-5,10)
ax.set_ylim(42,52)'''
plt.show()
```

3.4. Réaliser la carte suivante. Vous pouvez vous inspirer des indications de code ci-dessous

```
france = gpd.read_file("departements.geojson")
france_map=#dataframe à définir avec une jointure
#On convertit en geodataframe :
france_map = gpd.GeoDataFrame(france_map)
fig, ax = plt.subplots(1, figsize=(9,8))
france_map.plot(ax=ax,
column='DENS',
#https://matplotlib.org/stable/tutorials/colors/colormaps.html :
cmap='GnBu',
#crée des classes de densités avec effectifs égaux (mapclassify)
scheme='quantiles',
k=5, #nb de classes
legend=True,legend_kwds={'loc':'center left',
                         'bbox_to_anchor':(-0.1, 0.5)})
```



3.5. Réaliser la carte suivante représentant les 7 villes avec la population la plus importante (la taille du point est passée avec le paramètre `markersize` en ayant ici fait le choix de diviser par 100 la population extraite du dataframe)

Pour afficher le nom des villes vous pourrez vous aider du code suivant :

```
geom_points=gpd.points_from_xy(calcul avec LONGI_DMS,
                               calcul avec LATI_DMS)
gdf_pcpx=gpd.GeoDataFrame(...,geometry=geom_points)
for x, y, label in zip(gdf_pcpx.geometry.x, gdf_pcpx.geometry.y,
                               gdf_pcpx.NOM):
    ax.annotate(label, xy=(x, y), xytext=(3, 3),
                textcoords="offset points")
```