Annexe : Programme Python

Commandes utilisées dans l'étude statistique univariée des variables :

```
import numpy as np
import pandas as pandas
import matplotlib.pyplot as plt
import seaborn as sns

from matplotlib.ticker import AutoMinorLocator
import matplotlib.ticker as ticker


#Importation des données sous forme de Database

jo = pandas.read_csv("C:/Users/sarab/OneDrive/Documents/JO.csv",sep=",")

jo.head()

nb_individus = jo.index.size
nb_variables = jo.columns.size
print("\nIl y a ", nb_individus, " individus et ", nb_variables, "variables.")


jo.info()

variables = jo.columns
```

Voici les commandes python pour déterminer les données du 13e individu, des individus 32 à 47 auxquels s'ajoute le 213e individu, les données de la 4e variable, puis les commandes pour déterminer la taille du 27e individu ; en s'adaptant aux indices de python, soit pour les individus qui vont de 0 à 271 115.

```
indiv_treize_eme = jo.iloc[13]
print("Les données du 13eme individu sont :\n", indiv_treize_eme,"\n")
var_quat_eme = jo.iloc[:,4]
var_quat_eme_bis = jo[variables[4]]
print("Les données de la 4eme variable sont :\n", var_quat_eme,"\n")
indiv_time_vingtsept_eme = jo.iat[27,4]
print("Le 27eme individu est venu au ",indiv_time_vingtsept_eme,"\n")
print("Les données correspondant aux variables de la 32eme personne à la 47eme,avec aussi la 2
```

```
 47eme,avec aussi la 213 eme personne sont :\n", jo.iloc[list(range(32,47))+[213]],"\n")
```

Commandes pour déterminer le type des variables, et répertorier les valeurs/modalités de chaque variable et leur nombre :

```
print(jo.dtypes)


for k in range(0,nb_variables):
    var_k=variables[k]
    var_k_eff=jo[var_k].value_counts()
    var_k_nb_val=var_k_eff.size
    print("Valeurs/Modalités prises par la variable",var_k,": \n",var_k_eff,"\n pour un total de :", var_k_nb_val,"distinctes.\n")

#On peut ordonner en triant par ordre alphabétique

print("Classement des valeurs/modalités par ordre alphabétique :")
print("Variable Medal :")
print(jo["Medal"].value_counts().sort_index())
```

Pour les variables réelles, classer leurs valeurs par classe, et déterminer leurs effectifs par classe :

```
# Sélectionnez les variables float64
float_variables = jo.select_dtypes(include=['float64'])
# Pour chaque variable float64
for column in float_variables:
    print("Variable:", column)
    n = len(jo[column])
    #Calcul du nombre de classes
    num_classes = int(np.ceil(1 + np.log2(n)))
    # Classes de même longueur
    classes = pandas.cut(jo[column], bins=num_classes)

    # Effectifs par classe
    class_counts = classes.value_counts().sort_index()
    print("Effectifs par classe:",class_counts)
```

Etude d'une variable qualitative ordinale, Medal :

```
# Variable qualitative ordinnale : Medal

frequences_medal = jo["Medal"].value_counts()
print("Effectifs associés:")
print(frequences_medal)

mode_medal = jo["Medal"].mode()
print("Mode de la série:", mode_medal)


    # Diagramme en bâtons
plt.figure(figsize=(8, 6))
jo["Medal"].value_counts().plot(kind='bar', color='lightgreen')
plt.title("Diagramme en bâtons - Médaille")
plt.xlabel("Médaille")
plt.ylabel("Fréquence")
plt.xticks(rotation=45)
plt.grid(axis='y')

    # Diagramme circulaire
plt.figure(figsize=(8, 8))
jo["Medal"].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title("Diagramme circulaire - Médaille")
plt.ylabel('')
plt.show()
```

Etude d'une variable quantitative discrète, Year :

```
# Variable quantitative discrète : Year

# Effectifs
frequences_year = jo["Year"].value_counts().sort_index()
print("Effectifs associés à l'année:")
print(frequences_year)

# Effectifs cumulés
effectifs_cumules_year = frequences_year.cumsum()
print("Effectifs cumulés associés à l'année:")
print(effectifs_cumules_year)

# Fréquences cumulées
frequences_cumulees_year = effectifs_cumules_year / len(jo["Year"])
print("Fréquences cumulées associées à l'année:")
print(frequences_cumulees_year)

# Diagramme en bâtons pour les effectifs
plt.figure(figsize=(10, 6))
frequences_year.plot(kind='bar', color='skyblue')
plt.title("Diagramme en bâtons - Effectifs par année")
plt.xlabel("Année")
plt.ylabel("Effectifs")
plt.grid(axis='y')
plt.show()
```

```
# Diagramme en bâtons pour les effectifs cumulés
plt.figure(figsize=(10, 6))
effectifs_cumules_year.plot(kind='bar', color='salmon')
plt.title("Diagramme en bâtons - Effectifs cumulés par année")
plt.xlabel("Année")
plt.ylabel("Effectifs cumulés")
plt.grid(axis='y')
plt.show()

# Fonction de répartition empirique
plt.figure(figsize=(10, 6))
plt.step(frequences_year.index, frequences_cumulees_year, where='post')
plt.title("Fonction de répartition empirique - Année")
plt.xlabel("Année")
plt.ylabel("Fréquence cumulée")
plt.grid()
plt.show()

# Statistiques d'ordre principaux
print("Minimum de l'année:", jo["Year"].min())
print("Maximum de l'année:", jo["Year"].max())

# Statistiques de tendance centrale
print("Médiane de l'année:", jo["Year"].median())
print("Mode de l'année:", jo["Year"].mode())

# Statistiques de dispersion
print("Écart-type de l'année:", jo["Year"].std())
print("Variance de l'année:", jo["Year"].var())
```

```
# Box-plot pour l'année
plt.figure(figsize=(8, 6))
jo["Year"].plot(kind='box', vert=False)
plt.title("Box-plot - Année")
plt.xlabel("Année")
plt.show()

# Détection des outliers
Q1_year = jo["Year"].quantile(0.25)
Q3_year = jo["Year"].quantile(0.75)
IQR_year = Q3_year - Q1_year
lower_bound_year = Q1_year - 1.5 * IQR_year
upper_bound_year = Q3_year + 1.5 * IQR_year
outliers_year = jo[(jo["Year"] < lower_bound_year) | (jo["Year"] > upper_bound_year)]
print("Nombre d'outliers pour l'année:", len(outliers_year))
```

Voici les commandes pour montrer l'évolution les médailles d'équipe en fonction des années de jeux olympiques, ici le code nous montre l'évolution de la Chine aux jeux olympiques d'hiver.

```
donnees = pd.read_csv("JO.csv")
donnees_medailles = donnees[['Team', 'Year', 'Season', 'Medal']]
donnees_medailles = donnees_medailles.dropna(subset=['Medal'])
donnees_medailles = donnees_medailles[donnees_medailles['Season'] == 'Winter']
comptage_medailles = donnees_medailles.groupby(['Team', 'Year']).size().reset_index(name='Nombre de Médailles')
def tracer_evolution_medailles(equipe):
    donnees_equipe = comptage_medailles[comptage_medailles['Team'] == equipe]
    plt.figure(figsize=(10, 5))
    plot = sns.lineplot(data=donnees_equipe, x='Year', y='Nombre de Médailles', marker='o')
    plt.title(f'Évolution du nombre de médailles pour {equipe}')
    plt.xlabel('Année')
    plt.ylabel('Nombre de Médailles')
    plt.grid(True)
    for x, y in zip(donnees_equipe['Year'], donnees_equipe['Nombre de Médailles']):
        plt.text(x, y, str(x), color='gray', fontsize=9, ha='center')
    plt.show()
tracer_evolution_medailles('China')
```

Voici les méthodes pour la partie sur la partie de l'étude des performances des équipes en fonction de si elles sont à domicile ou non.

```python
import pandas as pd
import matplotlib.pyplot as plt

jo = pd.read_csv("Statistiques des médailles des pays à domicile ou non.csv", skipinitialspace=True)

domicile = jo[jo["domicile"] == "domicile"]
etranger = jo[jo["domicile"] == "etranger"]
median_domicile = domicile["moyenne"].median()
median_etranger = etranger["moyenne"].median()

moyenne_domicile = domicile["moyenne"].mean()
moyenne_etranger = etranger["moyenne"].mean()
median_domicile = domicile["mediane"].mean()
median_etranger = etranger["mediane"].mean()

plt.figure(figsize=(12, 6))

plt.bar(['Domicile (moyenne)', 'Étranger (moyenne)'], [moyenne_domicile, moyenne_etranger], color='blue', alpha=0.5)

plt.bar(['Domicile (médiane)', 'Étranger (médiane)'], [median_domicile, median_etranger], color='red', alpha=0.5)

plt.title('Moyenne et médiane des résultats des équipes à domicile et à l\'étranger')
plt.xlabel('Lieu')
plt.ylabel('Score')
plt.legend(['Moyenne', 'Médiane'])
plt.show()
```

Nous avons décidé de faire un autre document contenant les informations dont nous avions besoin pour ces graphes, cela nous a facilité la tâche voici un extrait :

```
"pays","domicile","moyenne","mediane","moyenne_athletes"
"United States","domicile",178.28571428571428,190.0,597.4285714285714
"United States","etranger",141.82142857142858,146.5,488.0357142857143
"France","domicile",92.0,92.0,677.5
"France","etranger",41.39393939393939,36.0,322.2121212121212
"Greece","domicile",34.333333333333336,31.0,317.6666666666667
"Greece","etranger",2.21875,1.0,63.21875
"Great Britain","domicile",116.66666666666667,122.0,688.0
"Great Britain","etranger",41.34375,32.5,291.875
"Australia","domicile",181.0,181.0,762.0
"Australia","etranger",33.088235294117645,12.0,198.55882352941177
"Brazil","domicile",46.0,46.0,571.0
"Brazil","etranger",11.852941176470589,1.5,94.1470588235294
```

Voici le code pour un des autres diagrammes en bâton de la partie :

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Statistiques des médailles des pays à domicile ou non.csv")

moyenne_domicile = df[df["domicile"] == "domicile"]["moyenne_athletes"].mean()
moyenne_etranger = df[df["domicile"] == "etranger"]["moyenne_athletes"].mean()

fig, ax = plt.subplots()

bar_labels = ['À domicile', 'À l\'étranger']
moyennes = [moyenne_domicile, moyenne_etranger]

plt.bar(bar_labels, moyennes, color=['green', 'green'])

plt.ylabel('Nombre moyen d\'athlètes')
plt.title('Moyenne du nombre d\'athlètes à domicile et à l\'étranger')

plt.show()
```

Voici un code qui permet d’obtenir les données que nous souhaitons étdudier

```
medailles_domicile = []
medailles_etranger = []
annee = 1896
while annee <= 2016:
    jo_annee = jo[(jo['Year']==annee) & (jo['Team']=='France')]
    nb_athlete_france = len(jo_annee)
    nb_medailles_france = jo_annee['Medal'].count()
    pourcentage = (nb_medailles_france/nb_athlete_france)*100

    if country(annee) == 'France':
        medailles_domicile.append(nb_medailles_france)
    else:
        medailles_etranger.append(nb_medailles_france)


    if annee==1912:
        annee+=8
    elif annee==1936:
        annee+=12
    elif annee==1904 or annee==1906 or annee >= 1992:
        annee += 2
    else:
        annee += 4

moyenne_domicile = np.mean(medailles_domicile)
moyenne_etranger = np.mean(medailles_etranger)
mediane_domicile = np.median(medailles_domicile)
mediane_etranger = np.median(medailles_etranger)
ecart_type_domicile = np.std(medailles_domicile)
ecart_type_etranger = np.std(medailles_etranger)
```

Ensuite ici on a la méthode qui montre le diagramme en bâtons pour le nombre de participant chinois et américains au jeux olympique, avec la Chine en bleu et les Etats-Unis en rouge.

```
donnees = pd.read_csv("JO.csv")
donnees_filtered = donnees.dropna(subset=['Team', 'Games', 'Name'])
donnees_filtered = donnees_filtered[(donnees_filtered['Team'].isin(['China', 'USA'])) & (donnees_filtered['Season'] == 'Summer')]
athletes_per_games = donnees_filtered.groupby(['Games', 'Team'])['Name'].nunique().reset_index(name='Nombre d\'Athlètes')
plt.figure(figsize=(14, 7))
sns.barplot(x='Games', y='Nombre d\'Athlètes', hue='Team', data=athletes_per_games, palette={'China': 'blue', 'USA': 'red'})
plt.title('Nombre d\'Athlètes de la Chine et des États-Unis par Jeux Olympiques')
plt.xlabel('Jeux Olympiques')
plt.ylabel('Nombre d\'Athlètes')
plt.xticks(rotation=45)
plt.legend(title='Équipe')
plt.tight_layout()
plt.show()
```

Ici on a le code qui montre la pente de régression linéaire des équipes qui participent aux jeux olympiques

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

donnees = pd.read_csv("JO.csv")
donnees_summer = donnees[donnees['Season'] == 'Summer']
donnees_medailles = donnees_summer[['Team', 'Year', 'Medal']].dropna(subset=['Medal'])
medal_counts = donnees_medailles.groupby(['Team', 'Year']).size().reset_index(name='Medal Count')

def prepare_regression_data(data):
    data['Year'] = pd.to_numeric(data['Year'])
    return data.pivot(index='Year', columns='Team', values='Medal Count').fillna(0)
regression_data = prepare_regression_data(medal_counts)
for team in regression_data.columns:
    y = regression_data[team].values
    X = regression_data.index.values.reshape(-1, 1)
    reg = LinearRegression().fit(X, y)
    slope = reg.coef_[0]
    if np.isfinite(slope):
        results.append((team, slope))
sns.lmplot(x='Year', y='China', data=regression_data.reset_index(), height=5, aspect=2, line_kws={'color': 'red'})
plt.title(f"Régression linéaire pour {'China'}")
plt.xlabel("Année")
plt.ylabel("Médailles")
plt.show()
```

Commandes qui servent à l'étude des résultats des équipes

Régression linéaire multiple avec les variables âge et taille :

```
hommes = jo[jo['Sex'] == 'M']

nombre_medailles_or_par_homme = hommes.groupby('Name')['Medal'].apply(lambda x: x.eq('Gold').sum()).reset_index()

donnees_hommes = pd.merge(nombre_medailles_or_par_homme, hommes[['Name', 'Age', 'Height']], on='Name', how='inner')


X = donnees_hommes[['Age', 'Height']]  # Variables indépendantes : âge et taille
Y = donnees_hommes['Medal']  # Variable dépendante : nombre de médailles d'or par homme

X = sm.add_constant(X)

modele = sm.OLS(Y, X)

resultats_modele = modele.fit()

print(resultats_modele.summary())
```

Création de boxplots liant différents sports à l'âge des femmes qui performent le mieux :

```
sports_inclus = ['Football', 'Gymnastics', 'Swimming', 'Athletics']

hommes_or_filtre = jo[(jo['Sport'].isin(sports_inclus)) & (jo['Medal'] == 'Gold') & (jo['Sex'] == 'F')]

plt.figure(figsize=(10, 6))
sns.boxplot(data=hommes_or_filtre, x="Sport", y="Age")

plt.title("Distribution de l'âge des femmes médaillées d'or par type de sport")
plt.xlabel("Type de sport")
plt.ylabel("Âge")

description = "Ce boxplot montre la distribution de l'âge des hommes médaillés d'or dans les sports suivants : Football, Gymnasti
plt.figtext(0.5, -0.1, description, wrap=True, horizontalalignment='center', fontsize=10)

plt.show()
```