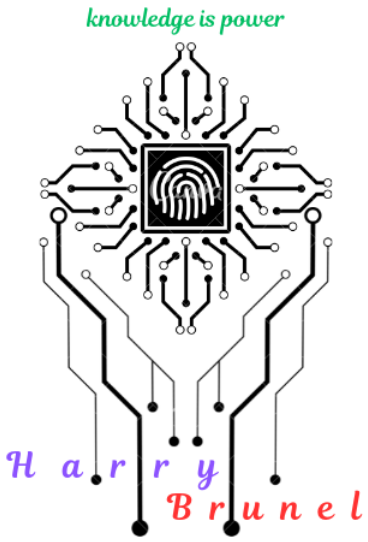



La donnée est l'actif stratégique de la révolution numérique

# Analyse budget et consommation avec Python

> - Source des données : ---
> </blocquote>

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Chargemebnt des données

budget = pd.read_csv("Budget.csv", sep=";")
comptes = pd.read_excel("Comptes.xlsx")
conso2014 = pd.read_csv("Conso2014.csv", sep=";")
conso2015 = pd.read_csv("Conso2015.csv", sep=";")
conso2016 = pd.read_csv("Conso2016.csv", sep=";")
conso2017 = pd.read_csv("Conso2017.csv", sep=";")
scenario = pd.read_excel("Scenario.xlsx")
services = pd.read_excel("Services.xlsx")
sites = pd.read_excel("Sites.xlsx")
```

In [3]:
```
# dictionnaire pour renommer les colonnes et corriger les noms de départements
dict_rename = {"Site": "CodeInsee", "Compte": "AccountKey" , "Année": "Annee", "Scenario": "S
               "Service": "DepartmentGroupKey"}
dict_replace = {"10" : "Aube", "11": "Aude", "13": "Bouches-du-Rhône"}
```

In [4]:
```
# Pour chaque data, changer les colonnes si possible

budget.rename(columns=dict_rename, inplace=True)

comptes.rename(columns=dict_rename, inplace=True)

conso2014.rename(columns=dict_rename, inplace=True)

conso2015.rename(columns=dict_rename, inplace=True)

conso2016.rename(columns=dict_rename, inplace=True)

conso2017.rename(columns=dict_rename, inplace=True)
```

> On remarque que certains codes pastaux ont quatre chiffres. Après vérication, ce sont les codes postaux qui commencent par zero. le fait de les enregistrer comme "int" agis sur ces chiffres. On corrige dans ce cas </blocquote>

In [5]:
```
budget["CodeInsee"] = budget["CodeInsee"].astype(str)
budget["CodeInsee"] = budget["CodeInsee"].apply(lambda x: "0" + x if len(x) == 4 else x)
```

In [6]:
```
budget.sample(10)
```

Out[6]:

| | ScenarioKey | CodeInsee | DepartmentGroupKey | AccountKey | Date | Montant |
|---|---|---|---|---|---|---|
| **586** | 2 | 67482 | 1 | 73 | 01/02/2016 | 5000 |
| **1650** | 2 | 44109 | 6 | 80 | 01/05/2016 | 7500 |
| **3572** | 2 | 78358 | 6 | 73 | 01/09/2016 | 2200 |
| **874** | 2 | 06088 | 2 | 60 | 01/03/2016 | 101000 |
| **2380** | 2 | 78646 | 1 | 66 | 01/06/2016 | 4800 |
| **3832** | 2 | 67482 | 6 | 61 | 01/10/2016 | 26000 |
| **1134** | 2 | 78358 | 6 | 52 | 01/03/2016 | 55000 |
| **167** | 2 | 06088 | 6 | 81 | 01/01/2016 | 2800 |
| **3059** | 2 | 75056 | 1 | 77 | 01/08/2016 | 4800 |
| **3706** | 2 | 06088 | 2 | 73 | 01/10/2016 | 1100 |

> Vérifions les données manquantes et les doublons </blocquote>

In [7]:
```
manquant = budget.isnull().sum()
doublons = budget.duplicated().sum()


print(manquant, "\n" + "---------", "\n", doublons)
```

```
ScenarioKey           0
CodeInsee             0
DepartmentGroupKey    0
AccountKey            0
Date                  0
Montant               0
dtype: int64
---------
 2
```

In [8]:
```
budget.drop_duplicates(inplace=True)
budget.duplicated().sum()
```

Out[8]: 0

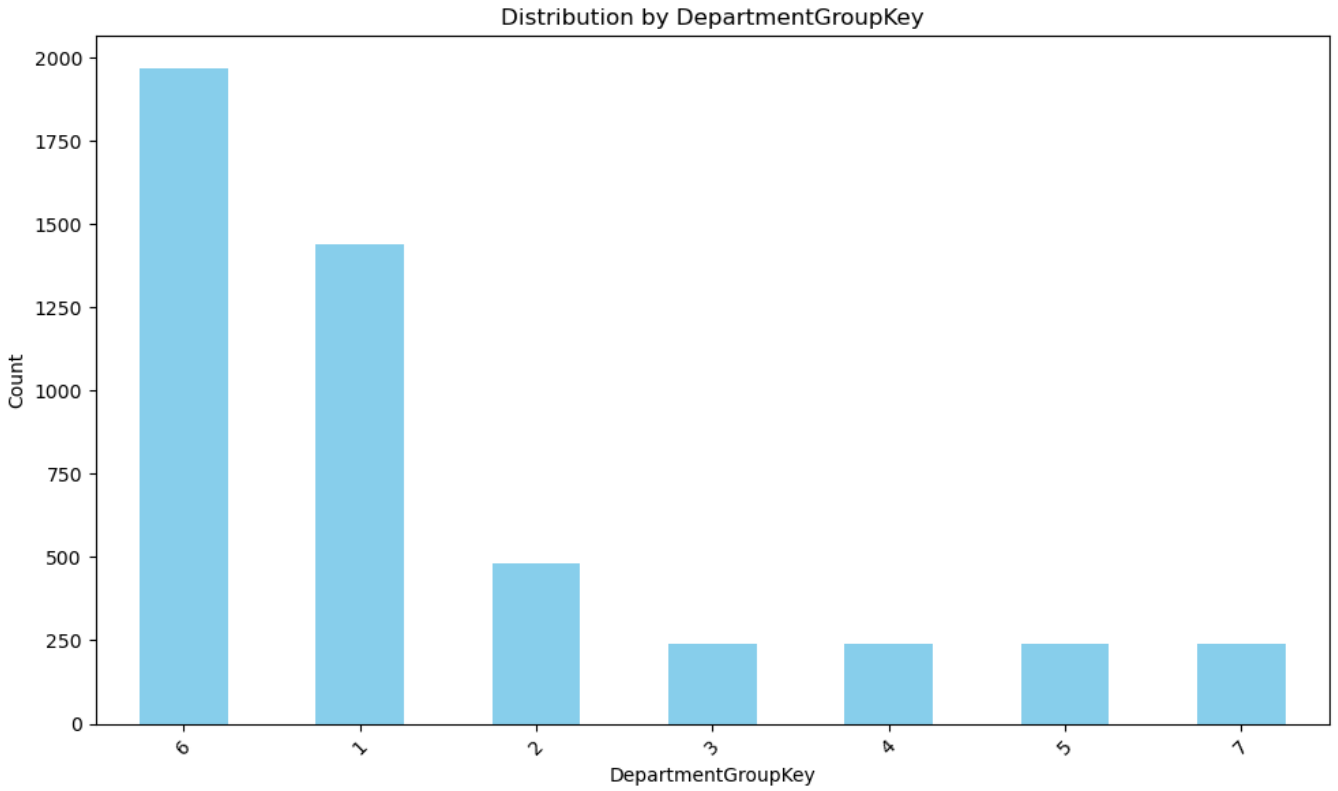
In [9]:
```
budget.describe()
```

Out[9]:

| | ScenarioKey | DepartmentGroupKey | AccountKey | Montant |
|---|---|---|---|---|
| **count** | 4844.0 | 4844.000000 | 4844.000000 | 4.844000e+03 |
| **mean** | 2.0 | 3.871594 | 72.959744 | 8.746448e+04 |
| **std** | 0.0 | 2.284988 | 9.935027 | 5.098863e+05 |
| **min** | 2.0 | 1.000000 | 52.000000 | -1.600000e+04 |
| **25%** | 2.0 | 1.000000 | 65.000000 | 1.000000e+03 |
| **50%** | 2.0 | 5.000000 | 74.000000 | 2.800000e+03 |
| **75%** | 2.0 | 6.000000 | 81.000000 | 7.500000e+03 |
| **max** | 2.0 | 7.000000 | 101.000000 | 8.917000e+06 |

In [10]:
```
#

departmentgroup_distribution = budget['DepartmentGroupKey'].value_counts()
```

In [11]:
```
# Function to plot distribution
def plot_distribution(data, title, xlabel, ylabel, kind="bar"):
    ax = data.plot(kind=kind, figsize=(10, 6), color='skyblue')
    ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
    plt.savefig("Distri_by_Department")

# Plot distributions

plot_distribution(departmentgroup_distribution, "Distribution by DepartmentGroupKey", "Departr
```



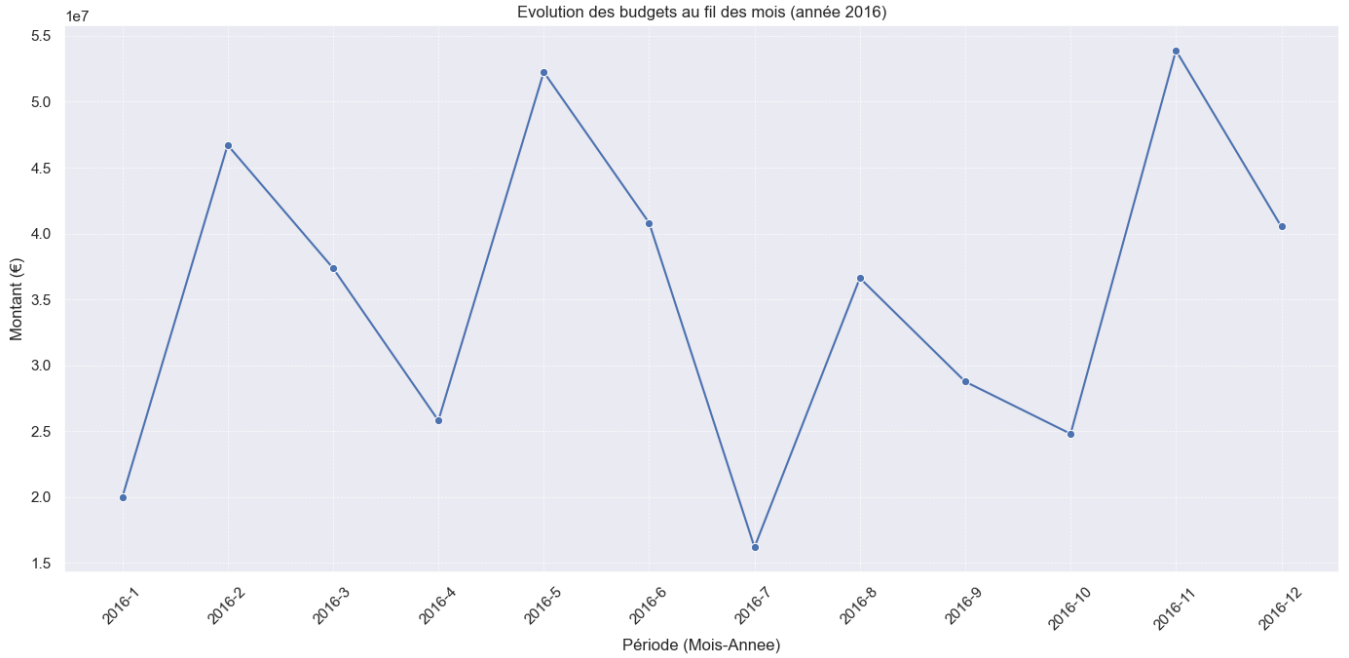

<Figure size 640x480 with 0 Axes>

In [12]:
```
# Convertion de 'Date' en datetime format
budget['Date'] = pd.to_datetime(budget['Date'], format='%d/%m/%Y')

# Extraction des mois et années
budget['Year'] = budget['Date'].dt.year
budget['Month'] = budget['Date'].dt.month

# Grouper par année et mois
time_series = budget.groupby(['Year', 'Month'])['Montant'].sum().reset_index()

#
time_series.head()
```

Out[12]:

| | Year | Month | Montant |
|---|---|---|---|
| **0** | 2016 | 1 | 19968750 |
| **1** | 2016 | 2 | 46714500 |
| **2** | 2016 | 3 | 37373500 |
| **3** | 2016 | 4 | 25796300 |
| **4** | 2016 | 5 | 52255300 |

In [13]:
```
# theme Seaborn  par défaut
sns.set_theme()

# Plotting the time series of expenses using Seaborn
plt.figure(figsize=(14, 7))
sns.lineplot(data=time_series, x=time_series['Year'].astype(str) + '-' + time_series['Month']
plt.title('Evolution des budgets au fil des mois (année 2016)')
plt.xlabel('Période (Mois-Annee)')
plt.ylabel('Montant (€)')
plt.xticks(rotation=45)
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
plt.savefig("evolution_budget")
```



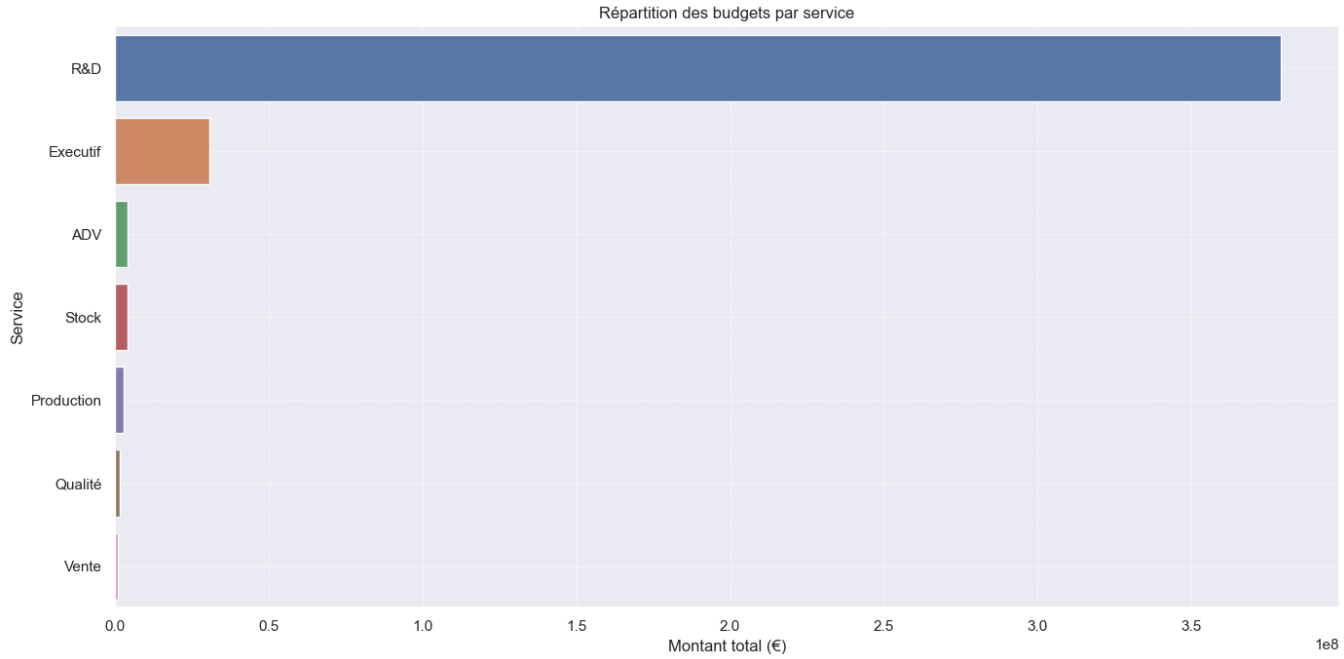

<Figure size 640x480 with 0 Axes>

In [14]:
```
# Convertion de 'DepartmentGroupKey'
budget['DepartmentGroupKey'] = budget['DepartmentGroupKey'].astype(str)
services['DepartmentGroupKey'] = services['DepartmentGroupKey'].astype(str)

# jointure des dataframes on 'DepartmentGroupKey'
budget_services = pd.merge(budget, services, on='DepartmentGroupKey', how='left')

# affichage
budget_services.head()
```

Out[14]:

| | ScenarioKey | CodeInsee | DepartmentGroupKey | AccountKey | Date | Montant | Year | Month | Service |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 44109 | 1 | 62 | 2016-01-01 | 14000 | 2016 | 1 | Executif |
| **1** | 2 | 44109 | 1 | 65 | 2016-01-01 | 3750 | 2016 | 1 | Executif |
| **2** | 2 | 44109 | 1 | 66 | 2016-01-01 | 5500 | 2016 | 1 | Executif |
| **3** | 2 | 44109 | 1 | 67 | 2016-01-01 | 2800 | 2016 | 1 | Executif |
| **4** | 2 | 44109 | 1 | 68 | 2016-01-01 | 2000 | 2016 | 1 | Executif |

In [15]:
```
# Regrouper par service et obtenir la somme des "Montant" pour chaque service
grouped_service = budget_services.groupby('Service')['Montant'].sum().sort_values(ascending=Fa

# Créer un diagramme à barres à l'aide de Seaborn
plt.figure(figsize=(14, 7))
sns.barplot(data=grouped_service, x='Montant', y='Service')
plt.title('Répartition des budgets par service')
plt.xlabel('Montant total (€)')
plt.ylabel('Service')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
plt.savefig("budget_par_services")
```




```
<Figure size 640x480 with 0 Axes>
```

In [16]:
```
all_conso = pd.concat([conso2014, conso2015, conso2016, conso2017], ignore_index=True)
all_conso.sample(10)
all_conso.to_csv("all_conso.csv")
```

In [17]:
```
data = pd.read_csv('all_conso.csv')
data.head()
```

Out[17]:

| | Unnamed: 0 | ScenarioKey | CodeInsee | DepartmentGroupKey | AccountKey | Annee | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 6088 | 1 | 60 | 2014 | 74195 | 71273 | 69758 | 87116 |
| **1** | 1 | 1 | 6088 | 1 | 61 | 2014 | 6573 | 7930 | 7333 | 8579 |
| **2** | 2 | 1 | 6088 | 1 | 62 | 2014 | 5548 | 5243 | 5159 | 7736 |
| **3** | 3 | 1 | 6088 | 1 | 65 | 2014 | 957 | 963 | 907 | 1328 |
| **4** | 4 | 1 | 6088 | 1 | 66 | 2014 | 871 | 876 | 825 | 1206 |

In [18]:
```
data.drop("Unnamed: 0", axis=1 , inplace=True)
# Afficher les premières lignes pour avoir un aperçu

data["CodeInsee"] = data["CodeInsee"].astype(str)
data["CodeInsee"] = data["CodeInsee"].apply(lambda x: "0" + x if len(x) == 4 else x).astype(st

data.head()
```

Out[18]:

| | ScenarioKey | CodeInsee | DepartmentGroupKey | AccountKey | Annee | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 06088 | 1 | 60 | 2014 | 74195 | 71273 | 69758 | 87116 | 74195 | 10 |
| **1** | 1 | 06088 | 1 | 61 | 2014 | 6573 | 7930 | 7333 | 8579 | 6573 | 1 |
| **2** | 1 | 06088 | 1 | 62 | 2014 | 5548 | 5243 | 5159 | 7736 | 5548 | |
| **3** | 1 | 06088 | 1 | 65 | 2014 | 957 | 963 | 907 | 1328 | 957 | |
| **4** | 1 | 06088 | 1 | 66 | 2014 | 871 | 876 | 825 | 1206 | 871 | |

In [19]:
```
# Remplacer les virgules par des points et convertir en float
for month in range(1, 13):
    data[str(month)] = data[str(month)].str.replace(',', '.').astype(float)

# Vérifier les changements en affichant à nouveau les premières lignes
data.head()
```

Out[19]:

| | ScenarioKey | CodeInsee | DepartmentGroupKey | AccountKey | Annee | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 06088 | 1 | 60 | 2014 | 74195.0 | 71273.0 | 69758.0 | 87116.0 | 741 |
| **1** | 1 | 06088 | 1 | 61 | 2014 | 6573.0 | 7930.0 | 7333.0 | 8579.0 | 65 |
| **2** | 1 | 06088 | 1 | 62 | 2014 | 5548.0 | 5243.0 | 5159.0 | 7736.0 | 55 |
| **3** | 1 | 06088 | 1 | 65 | 2014 | 957.0 | 963.0 | 907.0 | 1328.0 | 9 |
| **4** | 1 | 06088 | 1 | 66 | 2014 | 871.0 | 876.0 | 825.0 | 1206.0 | 8 |

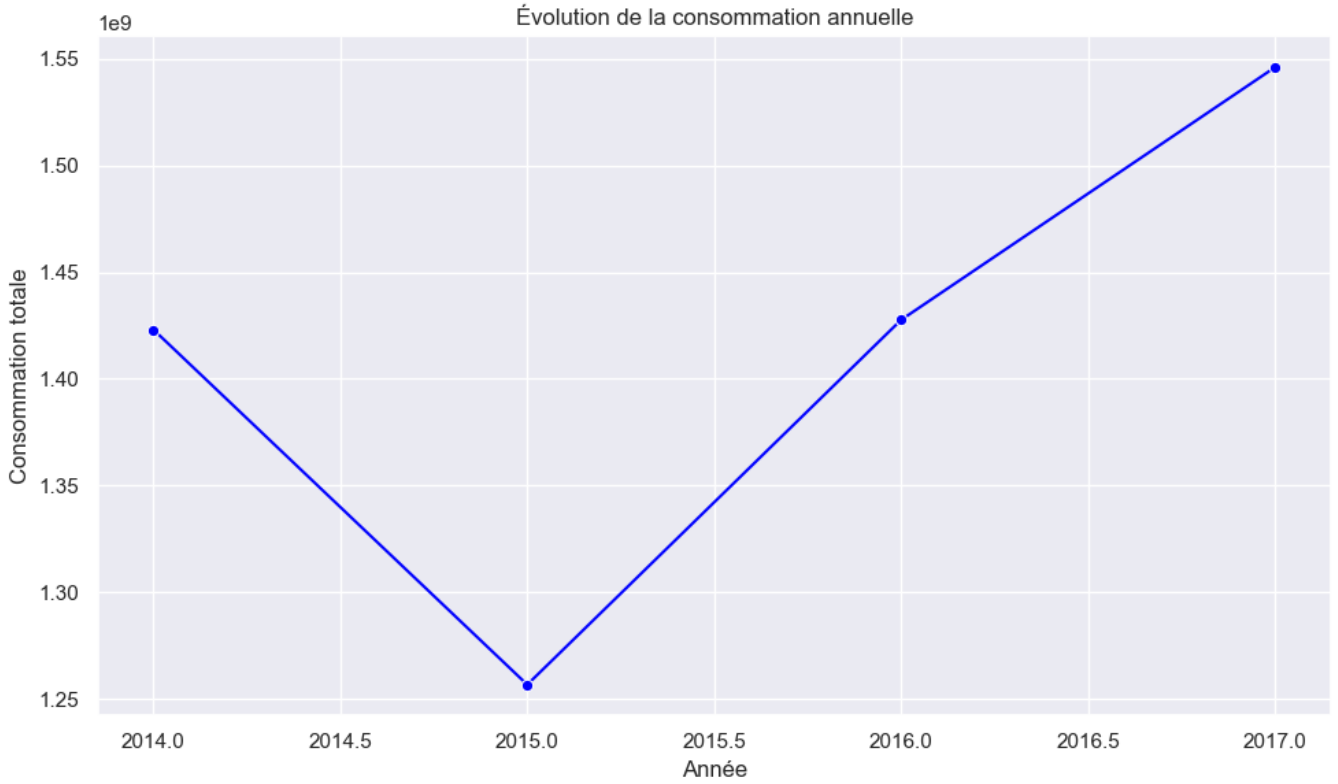
In [20]:
```
# Somme de la consommation pour chaque année
annual_consumption = data.groupby('Annee').sum().iloc[:, 5:17].sum(axis=1)

# Affichage de la consommation annuelle
annual_consumption
```

```
C:\Users\brune\AppData\Local\Temp\ipykernel_19184\1625472053.py:2: FutureWarning: The default
value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_onl
y will default to False. Either specify numeric_only or select only columns which should be v
alid for the function.
  annual_consumption = data.groupby('Annee').sum().iloc[:, 5:17].sum(axis=1)
```

Out[20]:

```
Annee
2014    1.423115e+09
2015    1.256621e+09
2016    1.427663e+09
2017    1.546113e+09
dtype: float64
```

In [21]:
```
# Créer la visualisation avec seaborn
plt.figure(figsize=(10, 6))
sns.lineplot(x=annual_consumption.index, y=annual_consumption, marker='o', color='blue')
plt.title("Évolution de la consommation annuelle")
plt.xlabel("Année")
plt.ylabel("Consommation totale")
plt.tight_layout()
plt.show()
plt.savefig("conso_par_an")
```




```
<Figure size 640x480 with 0 Axes>
```

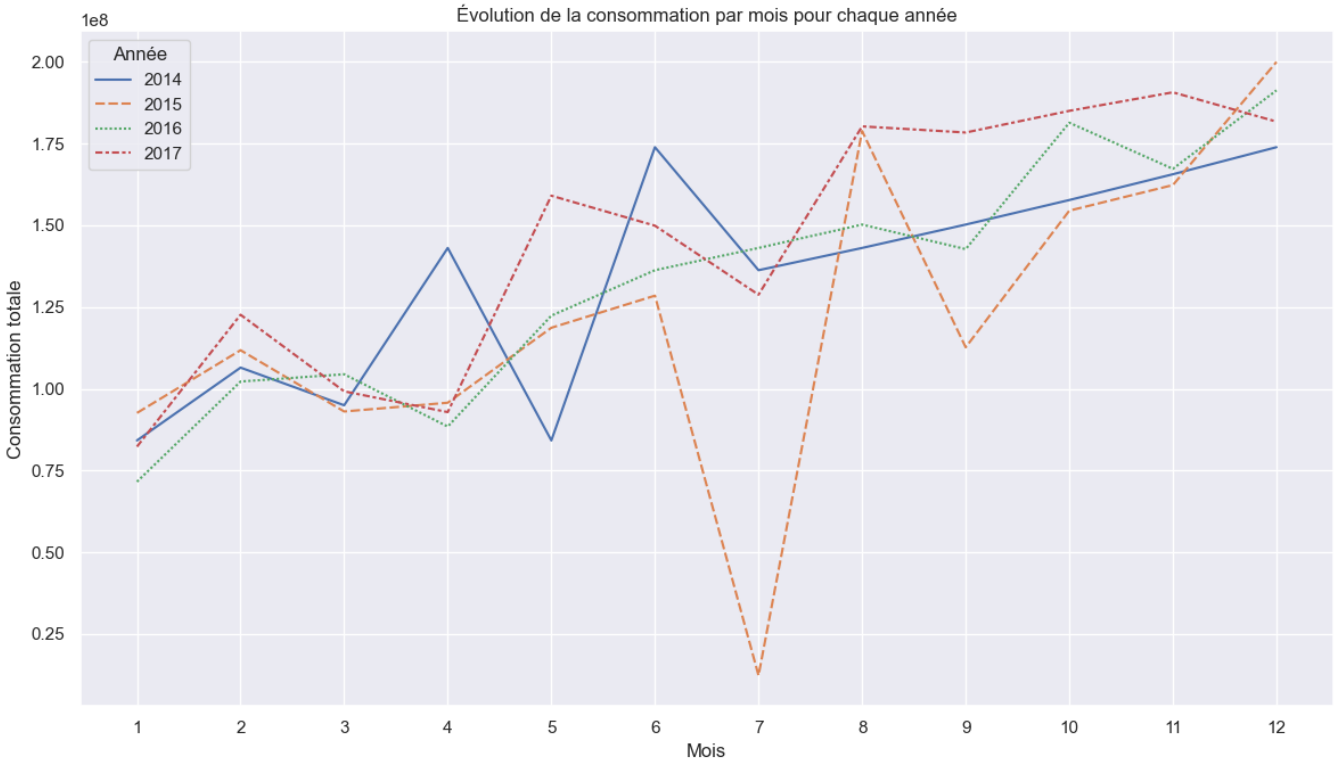
In [22]:
```
# Grouper par année et sommer les consommations pour chaque mois
monthly_consumption = data.groupby('Annee').sum().iloc[:, 3:16]

# Afficher les premières lignes pour vérifier
monthly_consumption.head()
```

```
C:\Users\brune\AppData\Local\Temp\ipykernel_19184\1073551443.py:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  monthly_consumption = data.groupby('Annee').sum().iloc[:, 3:16]
```

Out[22]:

| Annee | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|
| **2014** | 8.417750e+07 | 1.064754e+08 | 9.494336e+07 | 1.430944e+08 | 8.417750e+07 | 1.739322e+08 | 1.362804e+08 | 1. |
| **2015** | 9.259525e+07 | 1.117992e+08 | 9.304450e+07 | 9.572640e+07 | 1.186659e+08 | 1.284930e+08 | 1.226524e+07 | 1. |
| **2016** | 7.155088e+07 | 1.022164e+08 | 1.044377e+08 | 8.843296e+07 | 1.223742e+08 | 1.362804e+08 | 1.430944e+08 | 1. |
| **2017** | 8.228351e+07 | 1.226597e+08 | 9.921582e+07 | 9.285461e+07 | 1.590865e+08 | 1.499085e+08 | 1.287850e+08 | 1. |

In [23]:
```
# Tracer la visualisation de la consommation par mois pour chaque année
plt.figure(figsize=(12, 7))
sns.lineplot(data=monthly_consumption.T)
plt.title("Évolution de la consommation par mois pour chaque année")
plt.xlabel("Mois")
plt.ylabel("Consommation totale")
plt.legend(title='Année')
plt.tight_layout()
plt.show()
plt.savefig("conso_par_an_par_mois")
```




```
<Figure size 640x480 with 0 Axes>
```

In [24]:
```
data.iloc[:, 4:17]
```

Out[24]:

| | Annee | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2014 | 74195.0000 | 71273.000 | 69758.000 | 87116.0000 | 74195.000 | 105890.000 | 82968.000 | 871 |
| **1** | 2014 | 6573.0000 | 7930.000 | 7333.000 | 8579.0000 | 6573.000 | 10428.000 | 8171.000 | 85 |
| **2** | 2014 | 5548.0000 | 5243.000 | 5159.000 | 7736.0000 | 5548.000 | 9404.000 | 7368.000 | 77 |
| **3** | 2014 | 957.0000 | 963.000 | 907.000 | 1328.0000 | 957.000 | 1614.000 | 1265.000 | 13 |
| **4** | 2014 | 871.0000 | 876.000 | 825.000 | 1206.0000 | 871.000 | 1466.000 | 1149.000 | 12 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **4167** | 2017 | 119780.8950 | 178835.328 | 146900.875 | 145935.7725 | 179277.813 | 168934.920 | 145130.265 | 2031 |
| **4168** | 2017 | 21209.7950 | 25229.952 | 20583.365 | 21088.0425 | 25671.789 | 24190.320 | 20782.440 | 290 |
| **4169** | 2017 | 227482.8225 | 1030916.736 | -32953.030 | 118470.9015 | 1591765.461 | 1499932.665 | 1288578.375 | 18040 |
| **4170** | 2017 | 690245.9850 | 1039753.728 | 793761.100 | 605343.4380 | 1376382.150 | 1296976.065 | 1114220.205 | 15599 |
| **4171** | 2017 | 14733.8575 | 16647.552 | 13941.345 | 14836.4895 | 17990.973 | 16953.090 | 14564.340 | 203 |

4172 rows × 13 columns

In [25]:
```
# Vérifier le type de données de la colonne DepartmentGroupKey dans les deux dataframes
data_type_data = data["DepartmentGroupKey"].dtype
data_type_services = services["DepartmentGroupKey"].dtype

data_type_data, data_type_services
```

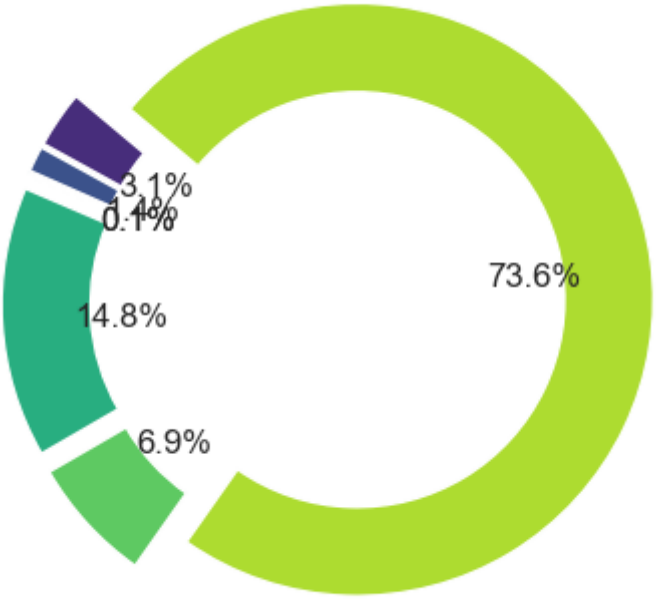
Out[25]:
```
(dtype('int64'), dtype('O'))
```

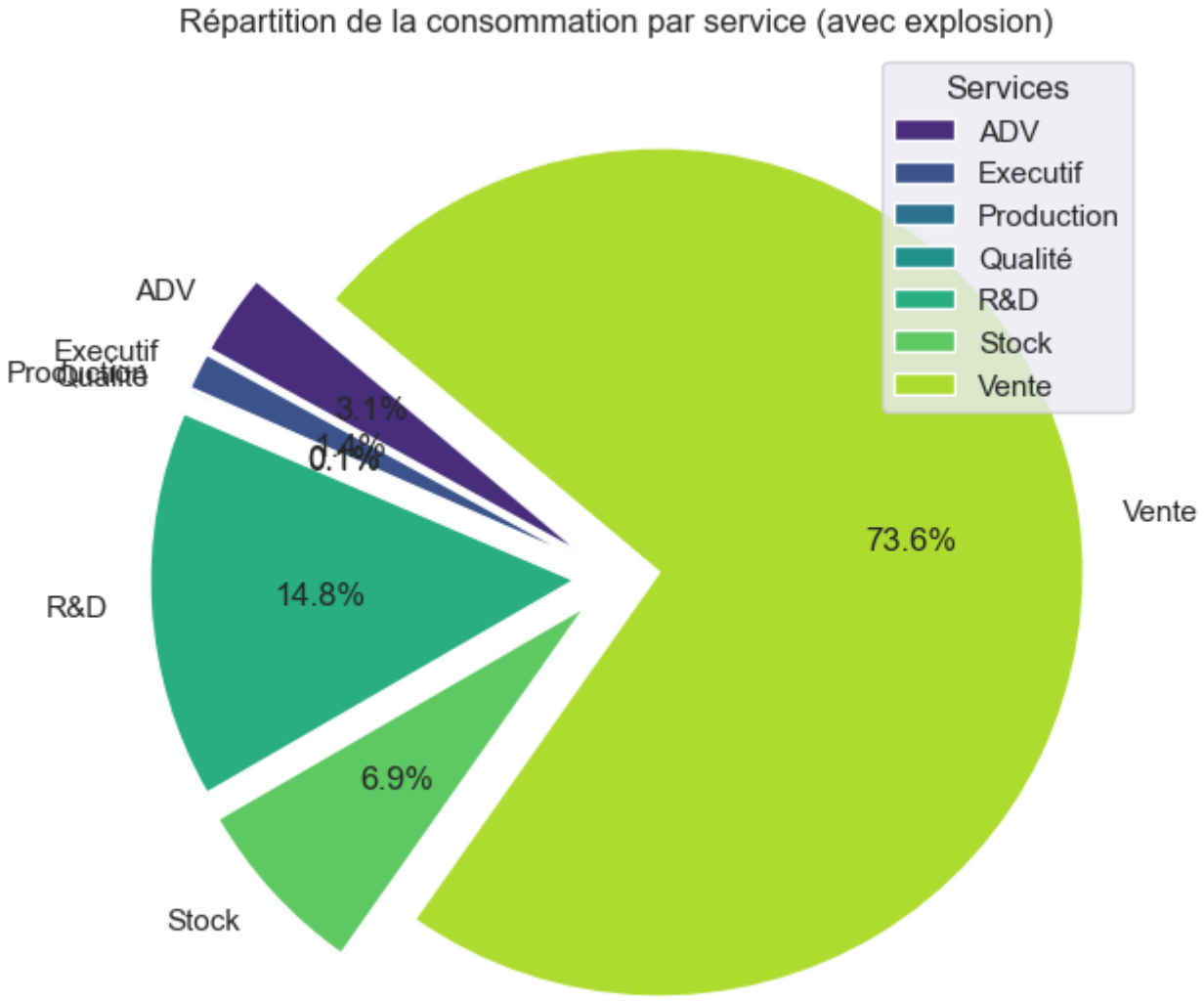
In [26]:
```
services["DepartmentGroupKey"] = services["DepartmentGroupKey"].astype(int)

# Fusionner data et services sur DepartmentGroupKey
merged_data = pd.merge(data, services, on="DepartmentGroupKey", how="left")

# Regrouper par Service et sommer les consommations pour chaque mois
consumption_by_service = merged_data.groupby('Service').sum().iloc[:, 3:17].sum(axis=1)

# Afficher la consommation par service
consumption_by_service
```

```
C:\Users\brune\AppData\Local\Temp\ipykernel_19184\4193213251.py:7: FutureWarning: The default
value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_onl
y will default to False. Either specify numeric_only or select only columns which should be v
alid for the function.
  consumption_by_service = merged_data.groupby('Service').sum().iloc[:, 3:17].sum(axis=1)
```

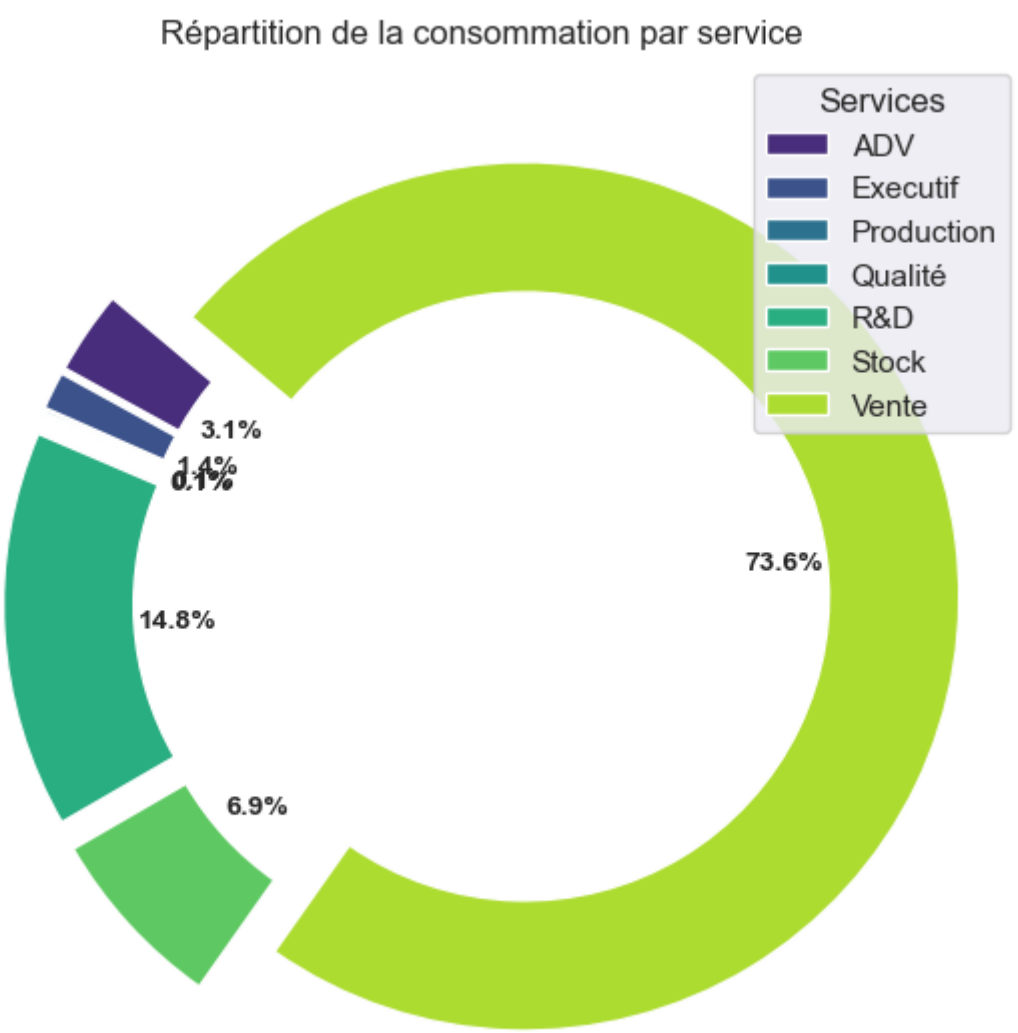
Out[26]:
```
Service
ADV           2.001121e+08
Executif      8.980124e+07
Production    7.139189e+06
Qualité       3.938523e+06
R&D           9.515590e+08
Stock         4.434074e+08
Vente         4.739721e+09
dtype: float64
```

In [27]:
```
# Créer un effet d'explosion pour toutes les sections
explode = [0.1] * len(consumption_by_service)
text_properties = {'fontsize': 12}
wedges, texts, autotexts = plt.pie(consumption_by_service, autopct='%1.1f%%', startangle=140,

# Diagramme en secteur avec l'effet explode
plt.figure(figsize=(7, 10))
plt.pie(consumption_by_service, labels=consumption_by_service.index, autopct='%1.1f%%', starta
plt.title("Répartition de la consommation par service (avec explosion)")
plt.legend(wedges, consumption_by_service.index, title="Services", loc="best")
plt.show()
plt.savefig("conso_service_diag_anneau")

# Diagramme en anneau pour la consommation par service avec une légende

plt.figure(figsize=(7, 10))
wedges, texts, autotexts = plt.pie(consumption_by_service, autopct='%1.1f%%', startangle=140,
plt.title("Répartition de la consommation par service")
plt.legend(wedges, consumption_by_service.index, title="Services", loc="best")
plt.setp(autotexts, size=10, weight="bold")
plt.show()
plt.savefig("conso_service_diag_secteur")
```




<Figure size 640x480 with 0 Axes>




```
<Figure size 640x480 with 0 Axes>
```

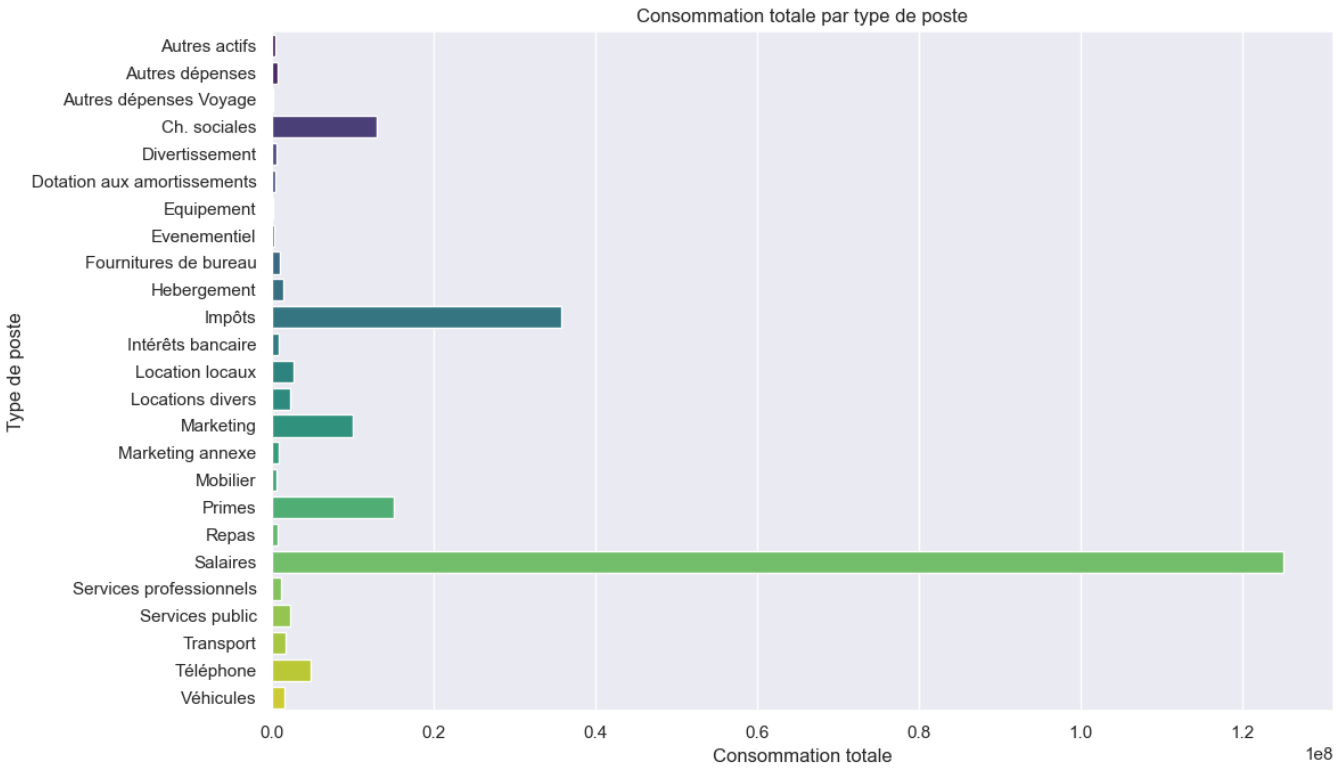
In [28]:

```
# Fusionner les dataframes data et comptes pour associer AccountKey à Poste
merged_data_poste = pd.merge(data, comptes, left_on="AccountKey", right_on="AccountKey", how='

# Regrouper par Poste et sommer les consommations pour chaque mois
consumption_by_poste = merged_data_poste.groupby('Poste').sum().iloc[:, 5:17].sum(axis=1)

# Afficher la consommation par poste
consumption_by_poste
```

```
C:\Users\brune\AppData\Local\Temp\ipykernel_19184\2204827171.py:5: FutureWarning: The default
value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_onl
y will default to False. Either specify numeric_only or select only columns which should be v
alid for the function.
  consumption_by_poste = merged_data_poste.groupby('Poste').sum().iloc[:, 5:17].sum(axis=1)
```

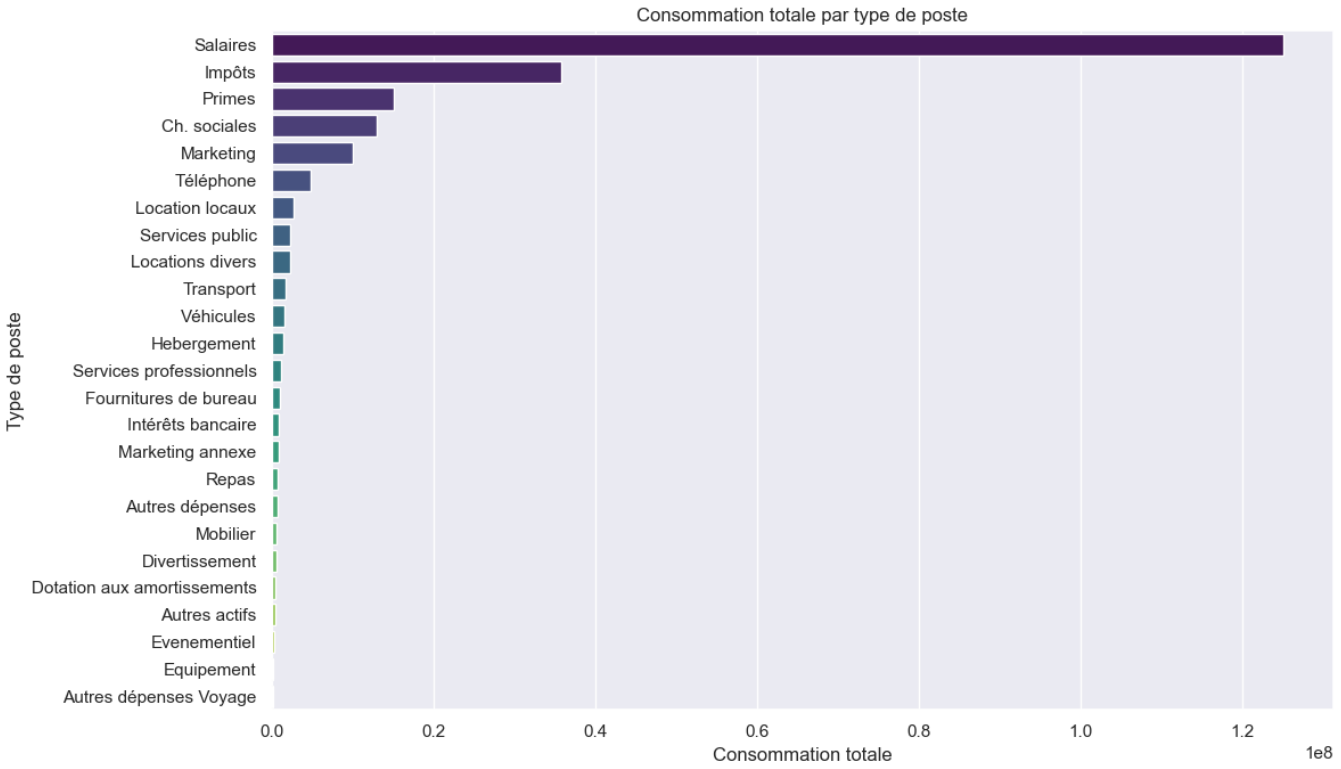
Out[28]:
```
Poste
Autres actifs                   5.141042e+05
Autres dépenses                 7.402522e+05
Autres dépenses Voyage          1.722141e+05
Ch. sociales                    1.294906e+07
Divertissement                  5.745881e+05
Dotation aux amortissements     5.174362e+05
Equipement                      1.797428e+05
Evenementiel                    2.854003e+05
Fournitures de bureau           1.092064e+06
Hebergement                     1.516300e+06
Impôts                          3.574340e+07
Intérêts bancaire               9.563583e+05
Location locaux                 2.674581e+06
Locations divers                2.271543e+06
Marketing                       1.006900e+07
Marketing annexe                9.545380e+05
Mobilier                        5.894804e+05
Primes                          1.506160e+07
Repas                           7.737750e+05
Salaires                        1.249898e+08
Services professionnels         1.197107e+06
Services public                 2.277650e+06
Transport                       1.684231e+06
Téléphone                       4.880293e+06
Véhicules                       1.634831e+06
dtype: float64
```

In [29]:
```
# Trier les valeurs par consommation
consumption_by_poste = consumption_by_poste
# Diagramme à barres horizontales
plt.figure(figsize=(12, 7))
sns.barplot(x=consumption_by_poste.values, y=consumption_by_poste.index, palette="viridis")
plt.title("Consommation totale par type de poste")
plt.xlabel("Consommation totale")
plt.ylabel("Type de poste")
plt.tight_layout()
plt.show()
plt.savefig("conso_poste")
```




```
<Figure size 640x480 with 0 Axes>
```

In [30]:
```
# Trier les valeurs par consommation
sorted_consumption_by_poste = consumption_by_poste.sort_values(ascending=False)

# Diagramme à barres horizontales
plt.figure(figsize=(12, 7))
sns.barplot(x=sorted_consumption_by_poste.values, y=sorted_consumption_by_poste.index, palette
plt.title("Consommation totale par type de poste")
plt.xlabel("Consommation totale")
plt.ylabel("Type de poste")
plt.tight_layout()
plt.show()
plt.savefig("conso_poste_trie")
```



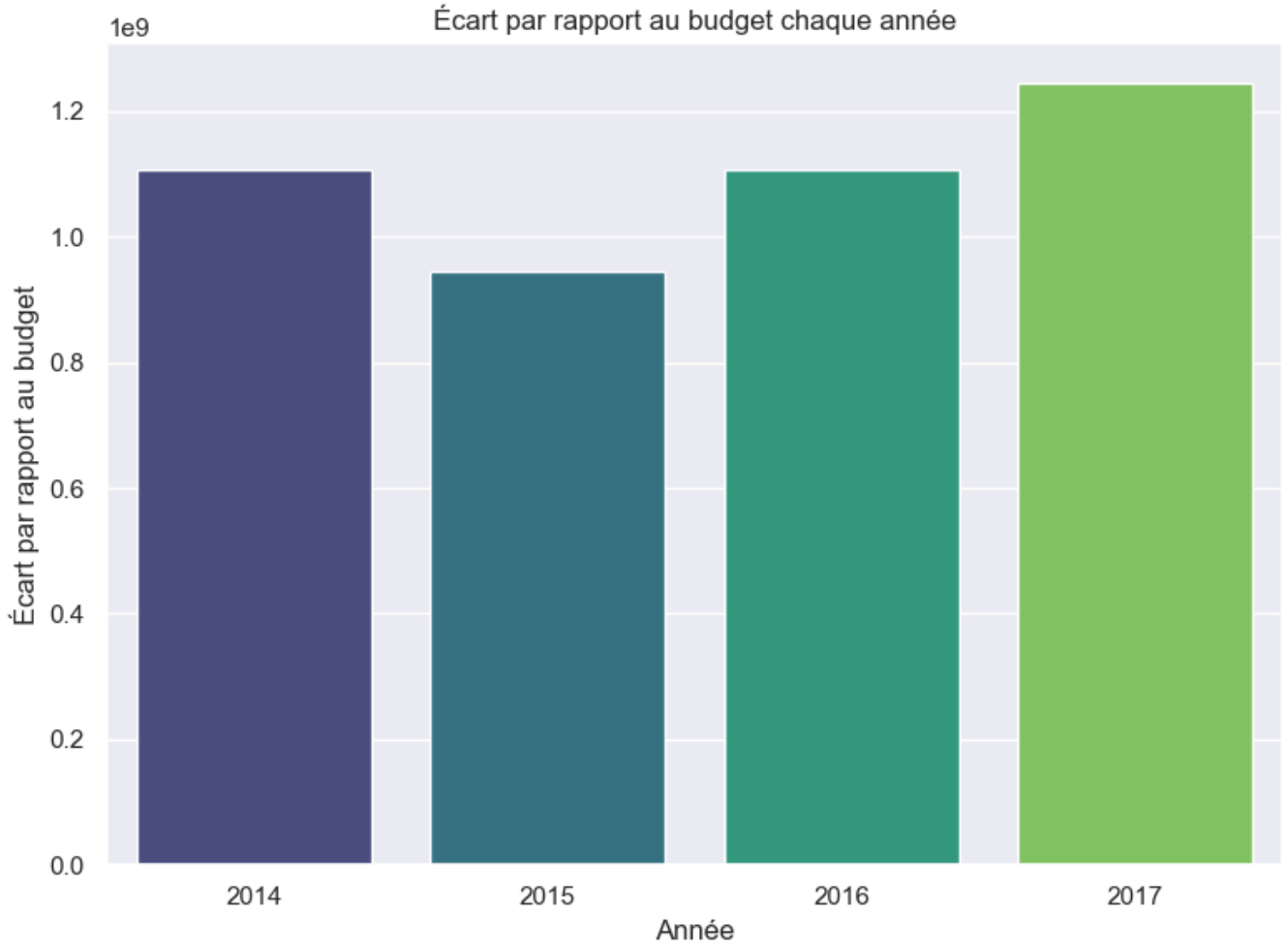

```
<Figure size 640x480 with 0 Axes>
```

In [31]:

```
import plotly.graph_objects as go
import kaleido
#!pip install kaleido

sites["CodeInsee"] = sites["CodeInsee"].astype(int)

data["CodeInsee"] = data["CodeInsee"].astype(int)

# Calculer la consommation totale en 2017
merged_data_departement = pd.merge(data, sites[['CodeInsee', 'Departement']], on="CodeInsee",
total_conso_2017 = merged_data_departement[merged_data_departement["Annee"] == 2017].iloc[:, 

# Créer la jauge
fig = go.Figure(go.Indicator(
    mode = "gauge+number",
    value = total_conso_2017,
    domain = {'x': [0, 1], 'y': [0, 1]},
    title = {'text': "Consommation totale en 2017"},
    gauge = {
        'axis': {'range': [None, 1.2 * total_conso_2017]},
        'steps': [
            {'range': [0, 0.6 * total_conso_2017], 'color': "lightgray"},
            {'range': [0.6 * total_conso_2017, 0.9 * total_conso_2017], 'color': "gray"},
        ],
        'threshold': {
            'line': {'color': "red", 'width': 4},
            'thickness': 0.75,
            'value': total_conso_2017}}))

fig.show()
#fig.write_image("conso_jauge.png")
```

In [32]:
```
# Calculer la consommation totale pour chaque année dans le dataframe data
consumption_per_year = data.groupby('Annee').sum().iloc[:, 5:17].sum(axis=1)

# Calculer le budget total pour chaque année dans le dataframe budget (en considérant que le b
total_budget = budget['Montant'].sum()

# Calculer l'écart entre la consommation et le budget pour chaque année
budget_difference = consumption_per_year - total_budget

# Afficher l'écart par rapport au budget pour chaque année
#budget_difference
```

In [33]:
```
# Diagramme à barres pour visualiser l'écart par rapport au budget chaque année
plt.figure(figsize=(8, 6))
sns.barplot(x=budget_difference.index, y=budget_difference.values, palette="viridis")
plt.title("Écart par rapport au budget chaque année")
plt.xlabel("Année")
plt.ylabel("Écart par rapport au budget")
plt.tight_layout()
plt.show()
```




In [34]:
```
# Modifier la colonne CodeInsee du dataframe sites
sites["CodeInsee"] = sites["CodeInsee"].astype(str)
sites["CodeInsee"] = sites["CodeInsee"].apply(lambda x: "0" + x if len(x) == 4 else x)

# Afficher les premières lignes pour vérifier les modifications
sites.sample(10)
```

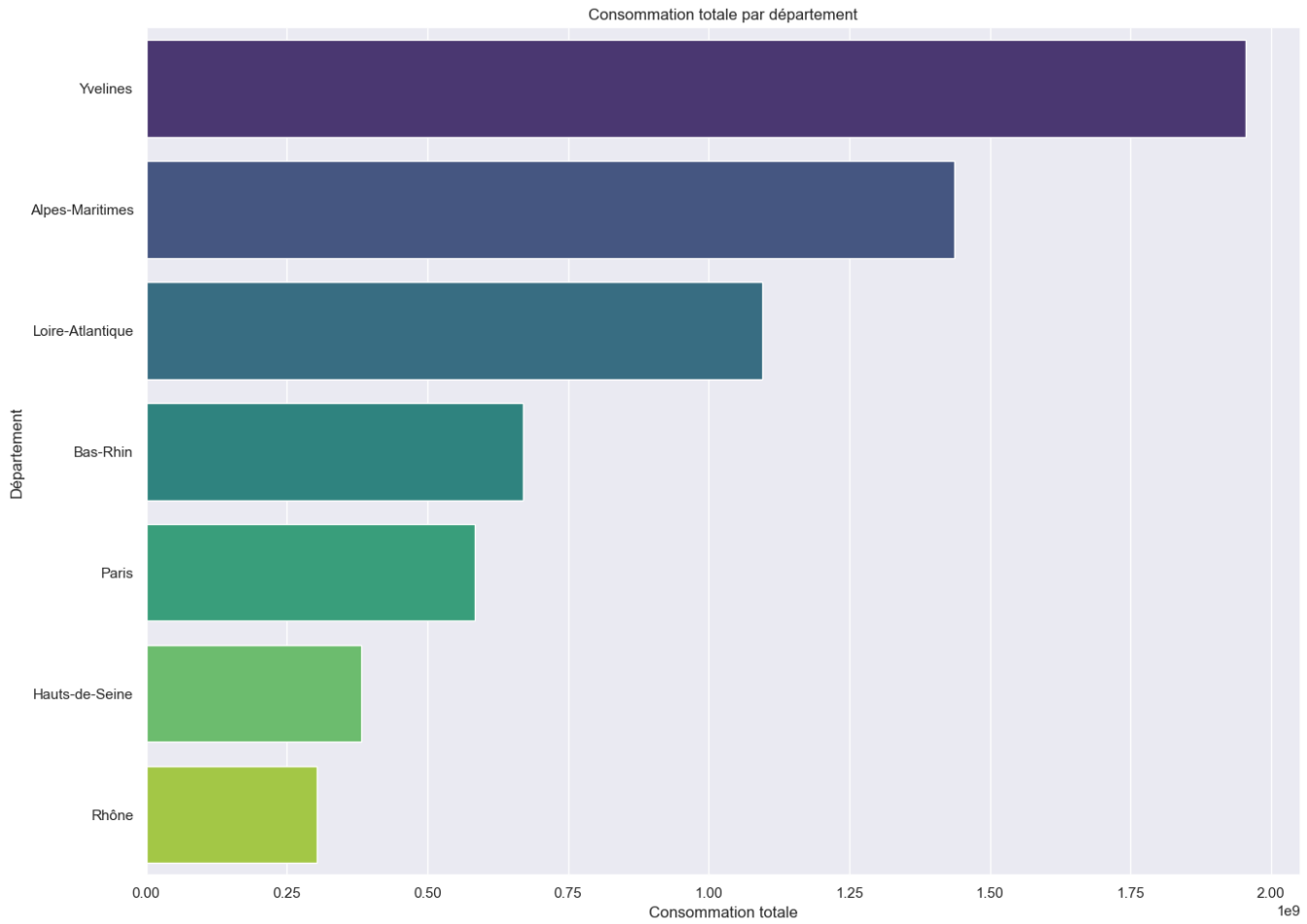
Out[34]:

| | CodeInsee | Commune | Departement | Region |
|---|---|---|---|---|
| **531** | 34003 | Agde | Hérault | Occitanie |
| **751** | 66037 | Canet-en-Roussillon | Pyrénées-Orientales | Occitanie |
| **374** | 10033 | Bar-sur-Aube | 10 | Grand Est |
| **14** | 77284 | Meaux | Seine-et-Marne | Ile de France |
| **691** | 57618 | Saint-Louis | Moselle | Grand Est |
| **137** | 93063 | Romainville | Seine-Saint-Denis | Ile de France |
| **716** | 61006 | Argentan | Orne | Normandie |
| **842** | 76447 | Montivilliers | Seine-Maritime | Normandie |
| **367** | 08409 | Sedan | Ardennes | Grand Est |
| **700** | 58102 | Donzy | Nièvre | Bourgogne-Franche-Comté |

In [35]:
```
sites["Departement"] = sites["Departement"].replace(dict_replace)
sites.Departement.unique()
```

Out[35]:
```
array(['Paris', 'Seine-et-Marne', 'Yvelines', 'Essone', 'Hauts-de-Seine',
       'Seine-Saint-Denis', 'Val-de-Marne', "Val d'oise", 'Nord',
       'Pas-de-Calais', 'Alpes-de-Haute-Provence', 'Hautes-Alpes',
       'Alpes-Maritimes', 13, 'Var', 'Vaucluse', 'Ain', 'Aisne', 'Allier',
       'Ardèche', 'Ardennes', 'Ariège', 10, 11, 12, 'Calvados', 'Cantal',
       'Charente', 'Charente-Maritime', 'Cher', 'Corrèze', "Côte-d'Or",
       "Côtes-d'Armor", 'Creuse', 'Dordogne', 'Doubs', 'Drôme', 'Eure',
       'Eure-et-Loir', 'Finistère', 'Gard', 'Haute-Garonne', 'Gers',
       'Hérault', 'Ille-et-Vilaine', 'Indre', 'Indre-et-Loire', 'Isère',
       'Jura', 'Landes', 'Loir-et-Cher', 'Loire', 'Haute-Loire',
       'Loire-Atlantique', 'Loiret', 'Lot', 'Lot-et-Garonne', 'Lozère',
       'Maine-et-Loire', 'Manche', 'Haute-Marne', 'Mayenne',
       'Meurthe-et-Moselle', 'Meuse', 'Morbihan', 'Moselle', 'Nièvre',
       'Oise', 'Orne', 'Puy-de-Dôme', 'Pyrénées-Atlantiques',
       'Hautes-Pyrénées', 'Pyrénées-Orientales', 'Bas-Rhin', 'Haut-Rhin',
       'Rhône', 'Haute-Saône', 'Saône-et-Loire', 'Sarthe', 'Savoie',
       'Haute-Savoie', 'Seine-Maritime', 'Deux-Sèvres', 'Somme', 'Tarn',
       'Tarn-et-Garonne', 'Vendée', 'Vienne', 'Haute-Vienne', 'Vosges',
       'Yonne', 'Territoire de Belfort'], dtype=object)
```

In [36]:
```
# Convertir la colonne CodeInsee du dataframe sites en int64
sites["CodeInsee"] = sites["CodeInsee"].astype(int)

data["CodeInsee"] = data["CodeInsee"].astype(int)

# Fusionner à nouveau les dataframes
merged_data_departement = pd.merge(data, sites[['CodeInsee', 'Departement']], on="CodeInsee",

# Regrouper par Departement et sommer les consommations pour chaque mois
consumption_by_departement = merged_data_departement.groupby('Departement').sum().iloc[:, 5:1

# Diagramme à barres pour visualiser la consommation par département
plt.figure(figsize=(14, 10))
sns.barplot(y=consumption_by_departement.index, x=consumption_by_departement.values, palette='
plt.title("Consommation totale par département")
plt.ylabel("Département")
plt.xlabel("Consommation totale")
plt.tight_layout()
plt.show()
```



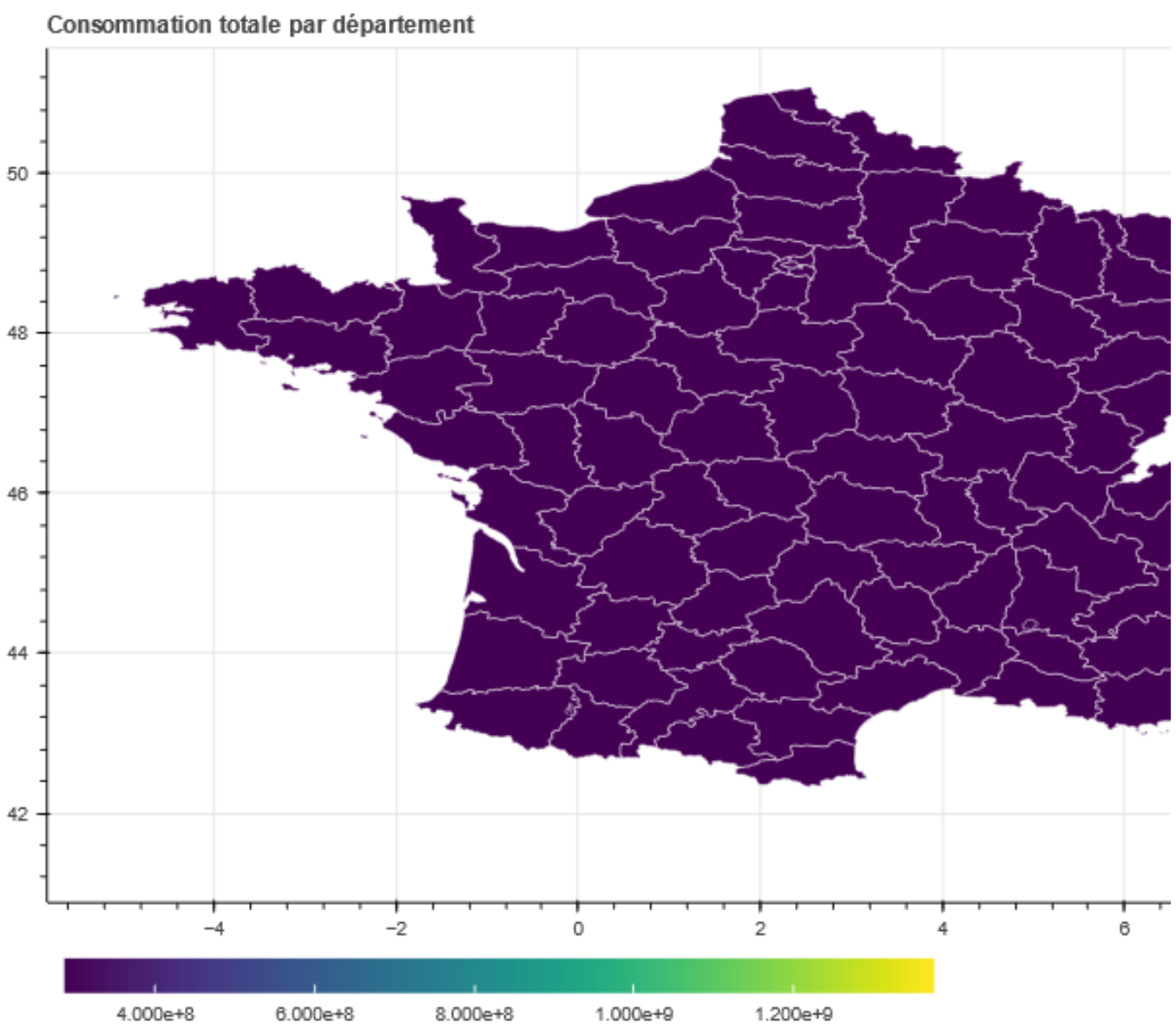

In [37]:
```
# Regrouper les données par CodeInsee et sommer la consommation pour chaque mois
consumption_by_codeinsee = merged_data_departement.groupby('CodeInsee').sum().iloc[:, 5:17].su

# Convertir le CodeInsee en string pour l'utiliser avec Plotly
consumption_by_codeinsee.index = consumption_by_codeinsee.index.astype(str)

# Vérifier les premières entrées des données préparées
consumption_by_codeinsee.head()
```

```
C:\Users\brune\AppData\Local\Temp\ipykernel_19184\2847228493.py:2: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
```

Out[37]:
```
CodeInsee
6088     1.376600e+09
44109    1.040910e+09
67482    6.293076e+08
69123    2.860163e+08
75056    5.462884e+08
dtype: float64
```

In [38]:
```
#!pip install bokeh
from bokeh.models import GeoJSONDataSource
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import push_notebook
from bokeh.palettes import Viridis256
from bokeh.models import ColorBar, LinearColorMapper
from bokeh.layouts import column

# Charger à nouveau le fichier geojson
with open("departements-version-simplifiee.geojson.txt", "r") as file:
    geojson_departements = json.load(file)
    min_val = consumption_by_codeinsee.min()
    max_val = consumption_by_codeinsee.max()
    color_mapper = LinearColorMapper(palette=Viridis256, low=min_val, high=max_val)

# Ajouter les valeurs de consommation dans les données GeoJSON
for feature in geojson_departements['features']:
    code = feature['properties']['code']
    if code in consumption_by_codeinsee:
        feature['properties']['consommation'] = consumption_by_codeinsee[code]
    else:
        feature['properties']['consommation'] = 0  # Si nous n'avons pas de données pour ce de

# Convertir le dataframe de consommation mis à jour en source de données pour Bokeh
source = GeoJSONDataSource(geojson=json.dumps(geojson_departements))

# Créer la figure avec la propriété correcte pour définir la hauteur
p = figure(title="Consommation totale par département",
           height=600,
           width=900,
           tools="pan,reset,save,wheel_zoom,box_zoom")

# Ajouter les polygones représentant les départements
p.patches('xs', 'ys', source=source,
          fill_color={'field': 'consommation', 'transform': color_mapper},
          line_color='white', line_width=0.5)

# Ajouter une barre de couleur
color_bar = ColorBar(color_mapper=color_mapper, label_standoff=8, width=500, height=20, locati
p.add_layout(color_bar, 'below')

# Afficher la figure
output_notebook()
show(p, notebook_handle=True)
```

BokehJS 3.2.2 successfully loaded.




Out[38]: <Bokeh Notebook handle for **In[38]**>

In [ ]: