IMPORTATIONS

```
import numpy as np
```

```
import datetime
```

```
!pip install folium
```

```
[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: folium in c:\users\laetitia_deken\anaconda3\lib\site-packages (0.17.0)
Requirement already satisfied: branca>=0.6.0 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from folium) (0.7.2)
Requirement already satisfied: jinja2>=2.9 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from folium) (3.1.3)
Requirement already satisfied: numpy in c:\users\laetitia_deken\anaconda3\lib\site-packages (from folium) (1.26.4)
Requirement already satisfied: requests in c:\users\laetitia_deken\anaconda3\lib\site-packages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in c:\users\laetitia_deken\appdata\roaming\python\python311\site-packages (from folium) (2023.10.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from requests->folium) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\laetitia_deken\anaconda3\lib\site-packages (from requests->folium) (2024.2.2)
```

```
# Conversion des coordonnées Lambert 93 en WGS84 pour les colonnes Coordonnée X et Coordonnée Y
```

```
Requirement already satisfied: pyproj in c:\users\laetitia_deken\anaconda3\lib\site-packages (3.6.1)
Requirement already satisfied: certifi in c:\users\laetitia_deken\anaconda3\lib\site-packages (from pyproj) (2024.2.2)

[notice] A new release of pip is available: 24.0 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
# Afficher toutes les lignes et colonnes
```

OUVERTURE ET LECTURE DU DATASET

```
df_log = pd.read_csv("C:/Users/Laetitia_Deken/OneDrive/Bureau/Projet Logement Social 2023/logements_sociaux_2023_01.csv", encoding='utf-8', se
```

[7]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Code Post
0	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	CODEPOSTA
1	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	014(
2	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	014(
3	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	014(
4	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	014(

```
df_log = df_log.drop(0)
```

[8]:

	Indice de répétition	Type de voie	Nom de voie	Etage	Nom du Programme	Lieu Dit	QPV_CODE	QPV_LIBELLE	TYPECONST_CODE	TYPECONST_LIBELLE	Nombre de pièces	Surface habitable	Année de construction	pre m lo
	NaN	NaN	Impasse des Soyeux	0	NaN	NaN	2	Non	I	individuel	3	67	2005	
	NaN	NaN	Impasse des Soyeux	0	NaN	NaN	2	Non	I	individuel	4	79	2005	
	NaN	NaN	Impasse des Soyeux	0	NaN	NaN	2	Non	I	individuel	4	79	2005	
	NaN	NaN	Impasse des Soyeux	0	NaN	NaN	2	Non	I	individuel	4	79	2005	

	NaN	NaN	Impasse des Merles	0	NaN	NaN	2	Non	I	individuel	3	68	1990

```
# Description des statistiques des colonnes numériques du dataset principal •••
```

[8]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Co Pos
count	5324304	5324304	5324304	5324304	5324304	5324304	5324005	5324005	5324304	5324304	5324
unique	18	18	101	101	1256	1253	6	6	16837	16285	6(
top	11	Île-de-France	59	Nord	200054781	Métropole du Grand Paris	1	pleine propriété	31555	Toulouse	511
freq	1379893	1379893	279432	279432	901424	901424	4956782	4956782	51532	51532	40

DATA CLEANING

Types

[9]:
```
# Conversion de ces trois colonnes d'années en entiers (int), en gérant les erreurs
df_log['Année de construction'] = pd.to_numeric(df_log['Année de construction'], errors='coerce').astype('Int64')
df_log['Année de première mise en location'] = pd.to_numeric(df_log['Année de première mise en location'], errors='coerce').astype('Int64')
df_log['Année d'entrée dans le patrimoine'] = pd.to_numeric(df_log['Année d'entrée dans le patrimoine'], errors='coerce').astype('Int64')
```

[10]:
```
# Conversion des colonnes en numérique (cela forcera les valeurs incorrectes à NaN)
df_log['Coordonnée X'] = pd.to_numeric(df_log['Coordonnée X'], errors='coerce')
df_log['Coordonnée Y'] = pd.to_numeric(df_log['Coordonnée Y'], errors='coerce')
```

[11]:
```
df_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5324304 entries, 1 to 5324304
Data columns (total 71 columns):
 #   Column                                  Dtype
---  ------                                  -----
 0   REG_CODE                                object
 1   REG_LIBELLE                             object
 2   DEP_CODE                                object
 3   DEP_LIBELLE                             object
 4   EPCI_CODE                               object
 5   EPCI_LIBELLE                            object
 6   DROIT_CODE                              object
 7   DROIT_LIBELLE                           object
 8   DEPCOM_CODE                             object
 9   DEPCOM_LIBELLE                          object
 10  Code Postal                             object
 11  Numéro de voie                          object
 12  Indice de répétition                    object
 13  Type de voie                            object
 14  Nom de voie                             object
 15  Etage                                   object
 16  Nom du Programme                        object
 17  Lieu Dit                                object
 18  QPV_CODE                                object
 19  QPV_LIBELLE                             object
 20  TYPECONST_CODE                          object
 21  TYPECONST_LIBELLE                       object
 22  Nombre de pièces                        object
 23  Surface habitable                       object
 24  Année de construction                   Int64
 25  Année de première mise en location      Int64
 26  Année d'entrée dans le patrimoine       Int64
 27  Origine de l'entrée dans le patrimoine  object
 28  FINAN_CODE                              object
 29  FINAN_LIBELLE                           object
 30  Complément du financement               object
 31  Convention APL                          object
 32  Numéro de Convention                    object
 33  Date de convention                      object
 34  NEWLOGT_CODE                            object
 35  NEWLOGT_LIBELLE                         object
 36  Catégorie financement CUS               object
 37  Date du DPE                             object
 38  Etiquette DPE Energie                   object
 39  Etiquette DPE GES                       object
 40  Année expiration de la convention       object
 41  Alinéa SRU                              object
 42  Code segment patrimoine                 object
 43  Libellé gestion patrimoine              object
 44  PMR_CODE                                object
 45  PMR_LIBELLE                             object
 46  Libellé de la voie dans le référentiel INSEE  object
 47  Système de coordonnées utilisée         object
 48  Coordonnée X                            float64
 49  Coordonnée Y                            float64
 50  Code QPV                                object
 51  PLG_IRIS2022_CODE                       object
 52  PLG_IRIS2022_LIBELLE                    object
 53  Code ZUS                                object
 54  Code ZFU                                object
 55  Code QVA                                object
 56  Qualité de la géolocalisation de la voie  object
 57  Qualité de la géolocalisation du numéro  object
 58  Qualité de la géolocalisation           object
 59  Erreur maximum de positionnement        object
 60  Qualité du géocodage pour le QPV        object
 61  Qualité du géocodage pour l'IRIS        object
```

```
 62  Qualité du géocodage pour le ZUS                 object
 63  Qualité du géocodage pour le ZFU                 object
 64  Qualité du géocodage pour le QVA                 object
 65  Indicatrice de présence d'un QPV dans la commune  object
 66  Indicatrice de présence d'une ZUS                object
 67  COMIRIS                                          object
 68  Unité Urbaine base 2020                          object
 69  Aire d'attraction des villes base 2020           object
 70  Zone d'emploi base 2020                          object
dtypes: Int64(3), float64(2), object(66)
memory usage: 2.8+ GB
```

Valeurs manquantes

[12]:
```
# Compter les valeurs manquantes pour chaque colonne
val_manquantes = df_log.isnull().sum()
val_manquantes
```

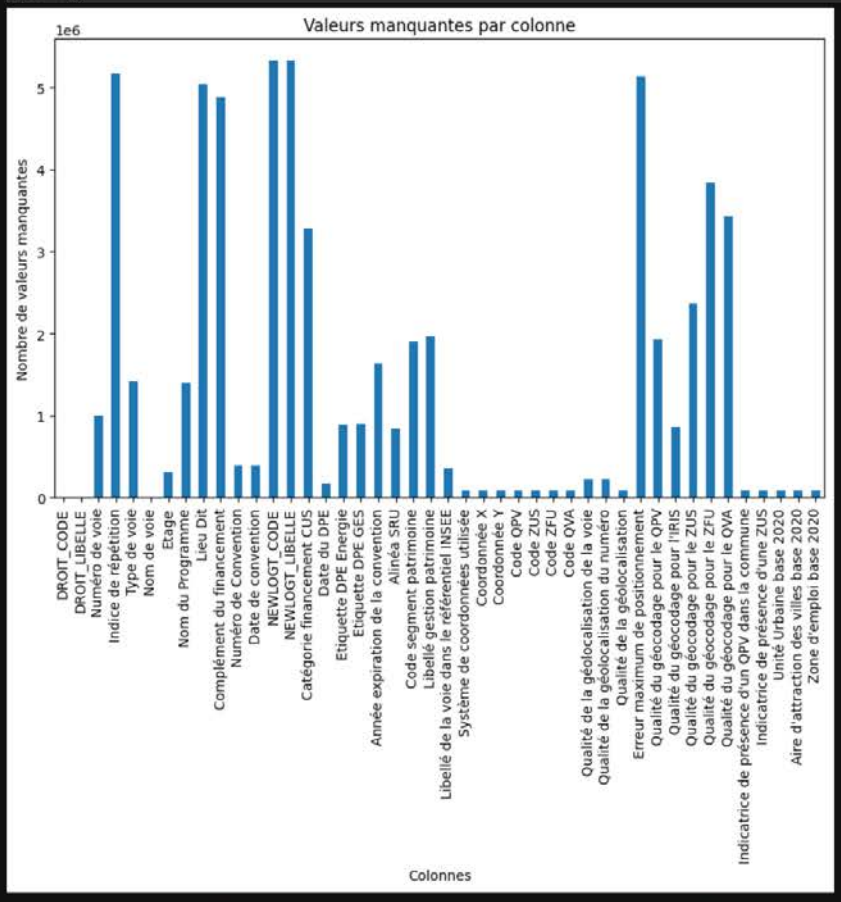
[12]:
```
REG_CODE                                        0
REG_LIBELLE                                     0
DEP_CODE                                        0
DEP_LIBELLE                                     0
EPCI_CODE                                       0
EPCI_LIBELLE                                    0
DROIT_CODE                                    299
DROIT_LIBELLE                                 299
DEPCOM_CODE                                     0
DEPCOM_LIBELLE                                  0
Code Postal                                     0
Numéro de voie                            1001323
Indice de répétition                      5176385
Type de voie                              1414567
Nom de voie                                  5293
Etage                                      301002
Nom du Programme                          1393354
Lieu Dit                                  5031948
QPV_CODE                                        0
QPV_LIBELLE                                     0
TYPECONST_CODE                                  0
TYPECONST_LIBELLE                               0
Nombre de pièces                                0
Surface habitable                               0
Année de construction                           0
Année de première mise en location              0
Année d'entrée dans le patrimoine               0
Origine de l'entrée dans le patrimoine          0
FINAN_CODE                                      0
FINAN_LIBELLE                                   0
Complément du financement                 4875334
Convention APL                                  0
Numéro de Convention                       393977
Date de convention                         394271
NEWLOGT_CODE                              5323786
NEWLOGT_LIBELLE                           5323786
Catégorie financement CUS                 3284061
Date du DPE                                176273
Etiquette DPE Energie                      885130
Etiquette DPE GES                          898990
Année expiration de la convention         1636140
Alinéa SRU                                 834988
Code segment patrimoine                   1904111
Libellé gestion patrimoine                1969461
PMR_CODE                                        0
PMR_LIBELLE                                     0
Libellé de la voie dans le référentiel INSEE    353707
Système de coordonnées utilisée                 83535
Coordonnée X                                    83535
Coordonnée Y                                    83535
Code QPV                                        83536
PLG_IRIS2022_CODE                               0
PLG_IRIS2022_LIBELLE                            0
Code ZUS                                        83673
Code ZFU                                        83682
Code QVA                                        83535
Qualité de la géolocalisation de la voie       221004
Qualité de la géolocalisation du numéro        221004
Qualité de la géolocalisation                   83535
Erreur maximum de positionnement          5133966
Qualité du géocodage pour le QPV          1926285
Qualité du géocodage pour l'IRIS           868952
Qualité du géocodage pour le ZUS          2362780
Qualité du géocodage pour le ZFU          3839498
Qualité du géocodage pour le QVA          3429973
Indicatrice de présence d'un QPV dans la commune      83535
Indicatrice de présence d'une ZUS               83535
COMIRIS                                             0
Unité Urbaine base 2020                         83535
Aire d'attraction des villes base 2020          83535
Zone d'emploi base 2020                         83535
dtype: int64
```

```
# Filtrer les colonnes qui possèdent des valeurs manquantes •••
```

[13]:
```
DROIT_CODE                                    299
DROIT_LIBELLE                                 299
Numéro de voie                            1001323
Indice de répétition                      5176385
Type de voie                              1414567
Nom de voie                                  5293
Etage                                      301002
Nom du Programme                          1393354
Lieu Dit                                  5031948
Complément du financement                 4875334
Numéro de Convention                       393977
Date de convention                         394271
NEWLOGT_CODE                              5323786
NEWLOGT_LIBELLE                           5323786
Catégorie financement CUS                 3284061
Date du DPE                                176273
Etiquette DPE Energie                      885130
Etiquette DPE GES                          898990
Année expiration de la convention         1636140
Alinéa SRU                                 834988
Code segment patrimoine                   1904111
Libellé gestion patrimoine                1969461
Libellé de la voie dans le référentiel INSEE    353707
Système de coordonnées utilisée                 83535
Coordonnée X                                    83535
```

```
Coordonnée Y                                    83535
Code QPV                                        83536
Code ZUS                                        83673
Code ZFU                                        83682
Code QVA                                        83535
Qualité de la géolocalisation de la voie       221004
Qualité de la géolocalisation du numéro        221004
Qualité de la géolocalisation                   83535
Erreur maximum de positionnement              5133966
Qualité du géocodage pour le QPV              1926285
Qualité du géocodage pour l'IRIS               868952
Qualité du géocodage pour le ZUS              2362780
Qualité du géocodage pour le ZFU              3839498
Qualité du géocodage pour le QVA              3429973
Indicatrice de présence d'un QPV dans la commune    83535
Indicatrice de présence d'une ZUS               83535
Unité Urbaine base 2020                         83535
Aire d'attraction des villes base 2020          83535
Zone d'emploi base 2020                         83535
dtype: int64
```

[14]:
```
# Visualiser les valeurs manquantes
plt.figure(figsize=(10, 6))
val_manquantes.plot(kind='bar')
plt.title('Valeurs manquantes par colonne')
plt.xlabel('Colonnes')
plt.ylabel('Nombre de valeurs manquantes')
plt.show()
```



Beaucoup de colonnes possèdent des valeurs manquantes mais qui ne sont pas dérangeantes dans le sens qu'elles ne nous serviront pas dans notre analyse.

[15]:
```
# Remplacement des valeurs manquantes ci-dessus dans les colonnes numériques par la médiane (voir describe)
for column in df_log.select_dtypes(include=['float64', 'int64']).columns:
    df_log[column] = df_log[column].fillna(df_log[column].median())
```

[16]:
```
# Remplacement des valeurs manquantes dans les colonnes non numériques par une chaîne de caractères vide
for column in df_log.select_dtypes(include=['object']).columns:
    df_log[column] = df_log[column].fillna('')
```

[17]:
```
df_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5324304 entries, 1 to 5324304
Data columns (total 71 columns):
 #   Column                                  Dtype
---  ------                                  -----
 0   REG_CODE                                object
 1   REG_LIBELLE                             object
 2   DEP_CODE                                object
 3   DEP_LIBELLE                             object
 4   EPCI_CODE                               object
 5   EPCI_LIBELLE                            object
 6   DROIT_CODE                              object
 7   DROIT_LIBELLE                           object
 8   DEPCOM_CODE                             object
 9   DEPCOM_LIBELLE                          object
 10  Code Postal                             object
 11  Numéro de voie                          object
 12  Indice de répétition                    object
 13  Type de voie                            object
 14  Nom de voie                             object
 15  Etage                                   object
 16  Nom du Programme                        object
 17  Lieu Dit                                object
 18  QPV_CODE                                object
```

```
 19  QPV_LIBELLE                                object
 20  TYPECONST_CODE                             object
 21  TYPECONST_LIBELLE                          object
 22  Nombre de pièces                           object
 23  Surface habitable                          object
 24  Année de construction                      Int64
 25  Année de première mise en location         Int64
 26  Année d'entrée dans le patrimoine          Int64
 27  Origine de l'entrée dans le patrimoine     object
 28  FINAN_CODE                                 object
 29  FINAN_LIBELLE                              object
 30  Complément du financement                  object
 31  Convention APL                             object
 32  Numéro de Convention                       object
 33  Date de convention                         object
 34  NEWLOGT_CODE                               object
 35  NEWLOGT_LIBELLE                            object
 36  Catégorie financement CUS                  object
 37  Date du DPE                                object
 38  Etiquette DPE Energie                      object
 39  Etiquette DPE GES                          object
 40  Année expiration de la convention          object
 41  Alinéa SRU                                 object
 42  Code segment patrimoine                    object
 43  Libellé gestion patrimoine                 object
 44  PMR_CODE                                   object
 45  PMR_LIBELLE                                object
 46  Libellé de la voie dans le référentiel INSEE  object
 47  Système de coordonnées utilisée            object
 48  Coordonnée X                               float64
 49  Coordonnée Y                               float64
 50  Code QPV                                   object
 51  PLG_IRIS2022_CODE                          object
 52  PLG_IRIS2022_LIBELLE                       object
 53  Code ZUS                                   object
 54  Code ZFU                                   object
 55  Code QVA                                   object
 56  Qualité de la géolocalisation de la voie   object
 57  Qualité de la géolocalisation du numéro    object
 58  Qualité de la géolocalisation              object
 59  Erreur maximum de positionnement           object
 60  Qualité du géocodage pour le QPV           object
 61  Qualité du géocodage pour l'IRIS           object
 62  Qualité du géocodage pour le ZUS           object
 63  Qualité du géocodage pour le ZFU           object
 64  Qualité du géocodage pour le QVA           object
 65  Indicatrice de présence d'un QPV dans la commune  object
 66  Indicatrice de présence d'une ZUS          object
 67  COMIRIS                                    object
 68  Unité Urbaine base 2020                    object
 69  Aire d'attraction des villes base 2020     object
 70  Zone d'emploi base 2020                    object
dtypes: Int64(3), float64(2), object(66)
memory usage: 2.8+ GB
```

DATA EXPLORATION

Evolution du parc social

[12]:
```
# Filtrer les données pour récupérer uniquement les Hauts-de-France
df_hdf = df_log[df_log['REG_LIBELLE'] == 'Hauts-de-France']
df_hdf.head()
```

[12]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Cod Postz
48875	32	Hauts-de-France	02	Aisne	200071785	CA Chauny-Tergnier-La Fère	1	pleine propriété	02001	Abbécourt	0230
48876	32	Hauts-de-France	02	Aisne	200071785	CA Chauny-Tergnier-La Fère	1	pleine propriété	02001	Abbécourt	0230
48877	32	Hauts-de-France	02	Aisne	240200477	CA GrandSoissons Agglomération	4	usufruit	02003	Acy	0220
48878	32	Hauts-de-France	02	Aisne	240200477	CA GrandSoissons Agglomération	4	usufruit	02003	Acy	0220
48879	32	Hauts-de-France	02	Aisne	240200477	CA GrandSoissons Agglomération	4	usufruit	02003	Acy	0220

[13]:
```
# Filtrer les données pour afficher uniquement la France métropolitaine (sans les DOM-TOM)
df_metropole = df_log[~df_log['REG_LIBELLE'].isin(['Guadeloupe', 'Martinique', 'Guyane', 'La Réunion', 'Mayotte'])] # ~ --> négation
df_metropole.head()
```

[13]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Code Postal	Nu de

1	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	01400
2	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	01400
3	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	01400
4	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	01400
5	84	Auvergne-Rhône-Alpes	01	Ain	200069193	CC de la Dombes	1	pleine propriété	01001	L'Abergement-Clémenciat	01400

[20]:
```
df_hdf.describe()
```

[20]:

	Année de construction	Année de première mise en location	Année d'entrée dans le patrimoine	Coordonnée X	Coordonnée Y
count	593655.0	593655.0	593655.0	593655.000000	5.936550e+05
mean	1977.56348	1980.257692	1981.720808	688482.352553	7.021159e+06
std	28.351809	26.839725	27.674457	37756.661242	5.776198e+04
min	1097.0	1850.0	1870.0	583950.000000	6.835944e+06
25%	1965.0	1967.0	1968.0	660815.230000	6.989350e+06
50%	1978.0	1980.0	1981.0	694398.800000	7.036876e+06
75%	1998.0	2002.0	2005.0	711630.430000	7.060677e+06
max	2022.0	2022.0	2022.0	786714.910000	7.109542e+06

[21]:
```
df_metropole.describe()
```

[21]:

	Année de construction	Année de première mise en location	Année d'entrée dans le patrimoine	Coordonnée X	Coordonnée Y
count	5143684.0	5143684.0	5143684.0	5.143684e+06	5.143684e+06
mean	1981.078692	1985.298796	1988.622801	6.793338e+05	6.719168e+06
std	25.726279	22.924213	23.242097	1.823929e+05	2.404624e+05
min	1010.0	1850.0	1850.0	1.017500e+05	6.051850e+06
25%	1966.0	1969.0	1971.0	5.998120e+05	6.522003e+06
50%	1978.0	1984.0	1989.0	6.574636e+05	6.835944e+06
75%	2001.0	2006.0	2010.0	8.112933e+05	6.871630e+06
max	2023.0	2023.0	2023.0	1.239177e+06	7.109542e+06

Le dataset contient plus de 5 millions de logements, dont 593 000 dans les Hauts-de-France, ce qui représente 10 % du total.

Ouverture initiale du Notebook jusqu'ici pour charger les données... :)

Nombre de logements sociaux construits chaque année en Hauts-de-France (HDF) et en France métropolitaine (FM)

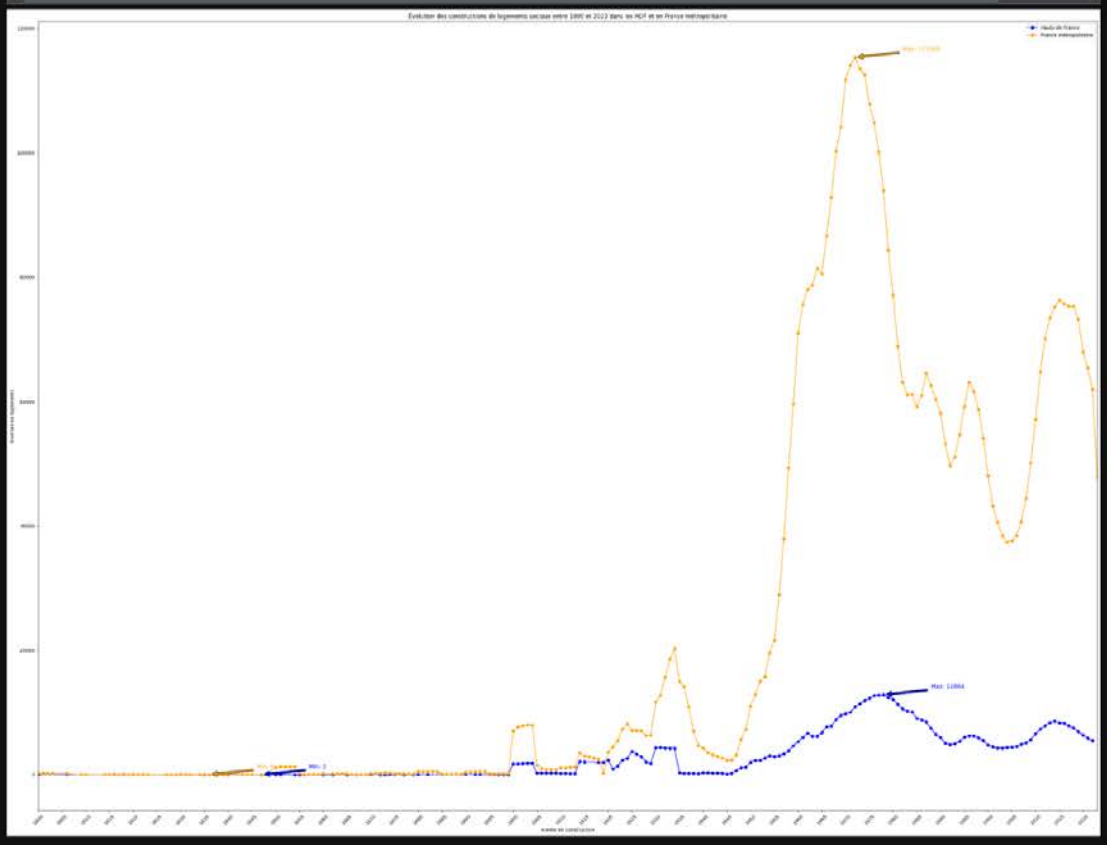
[20]:
```
# Groupement des données par année de construction pour calculer le nombre de logements construits chaque année
hdf_construction = df_hdf.groupby('Année de construction').size()
metropole_construction = df_metropole.groupby('Année de construction').size()
```

[21]:
```
# Ajout des courbes avec un style lissé (rolling mean)
hdf_smoothed = hdf_construction.rolling(window=5).mean()
metropole_smoothed = metropole_construction.rolling(window=5).mean()
```

[22]:
```
# Assurer que l'index est de type entier
hdf_smoothed.index = hdf_smoothed.index.astype(int)
metropole_smoothed.index = metropole_smoothed.index.astype(int)
```

[23]:
```
# Filtrage des années pour éviter les années aberrantes, on garde ici les années de 1800 à 2023 uniquement
hdf_filtre = hdf_smoothed[(hdf_smoothed.index >= 1800) & (hdf_smoothed.index <= 2023)]
metropole_filtre = metropole_smoothed[(metropole_smoothed.index >= 1800) & (metropole_smoothed.index <= 2023)]

# Création du graphique en courbes et de sa taille
plt.figure(figsize=(40, 30))

# Ajout des courbes avec un style lissé (rolling mean) sur les années filtrées
plt.plot(hdf_filtre.index, hdf_filtre.values, marker='o', color='blue', label='Hauts-de-France')
plt.plot(metropole_filtre.index, metropole_filtre.values, marker='o', color='orange', label='France métropolitaine')

plt.title('Évolution des constructions de logements sociaux entre 1800 et 2023 dans les HDF et en France métropolitaine')
plt.xlabel('Année de construction')
plt.ylabel('Nombre de logements')
plt.legend()
plt.grid(False)

# Ajout d'une étiquette pour les valeurs minimale et maximale des Hauts-de-France
hdf_min_annee = hdf_filtre.idxmin()
hdf_max_annee = hdf_filtre.idxmax()
```

```
plt.annotate(f'Min: {hdf_filtre.min():.0f}', xy=(hdf_min_annee, hdf_filtre.min()), xytext=(hdf_min_annee + 10, hdf_filtre.min() + 1000),
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=12, color='blue')
plt.annotate(f'Max: {hdf_filtre.max():.0f}', xy=(hdf_max_annee, hdf_filtre.max()), xytext=(hdf_max_annee + 10, hdf_filtre.max() + 1000),
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=12, color='blue')

# Ajout d'une étiquette pour les valeurs minimale et maximale de la France métropolitaine
metropole_min_annee = metropole_filtre.idxmin()
metropole_max_annee = metropole_filtre.idxmax()

plt.annotate(f'Min: {metropole_filtre.min():.0f}', xy=(metropole_min_annee, metropole_filtre.min()), xytext=(metropole_min_annee + 10, metropole_filtre.m
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=12, color='orange')
plt.annotate(f'Max: {metropole_filtre.max():.0f}', xy=(metropole_max_annee, metropole_filtre.max()), xytext=(metropole_max_annee + 10, metropole_filtre.m
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=12, color='orange')

# Rotation des labels sur l'axe X pour les rendre plus lisibles
plt.xticks(rotation=45)

# Affichage d'un label sur l'axe X tous les 5 ans
years = np.arange(1800, 2023, 5)
plt.xticks(years)

# Limitation des années pour éviter les années aberrantes
plt.xlim(1800, 2023)

# Sauvegarde et affichage du graphique
plt.savefig('C:/Users/Laetitia_Deken/OneDrive/Bureau/Projet Logement Social 2023/evolution_construction_annotated.png', dpi=300)
plt.show()
```



On peut constater une évolution marquée dans la construction des logements sociaux en France métropolitaine : en effet, la courbe orange montre une augmentation très marquée du nombre de logements construits, avec un pic très prononcé autour de certaines années (années 1970-1980), suivi d'une forte baisse. Ce pic pourrait correspondre à une période d'investissement intensif dans le logement social, souvent lié à des politiques publiques spécifiques à ce moment-là.

Une différence significative entre les Hauts-de-France et le reste de la France ressort également : la courbe des Hauts-de-France (en bleu) est bien en dessous de celle de la France métropolitaine (logique, il n'y a qu'une région ici), ce qui indique que cette région a eu un nombre de constructions nettement inférieur. Il y a aussi une stabilité relative dans la construction de logements sociaux dans les Hauts-de-France avec quelques fluctuations, mais sans atteindre les niveaux élevés observés au niveau national. On constate que les pics et baisses ont lieu dans les mêmes périodes, ce qui montre que les HDF ont suivi globalement la tendance nationale.

Stabilisation ou baisse récente : les deux courbes semblent montrer une stabilisation ou même une baisse dans les constructions récentes. Cette tendance pourrait être due à une saturation du marché, à une modification des priorités politiques, ou à des contraintes économiques.

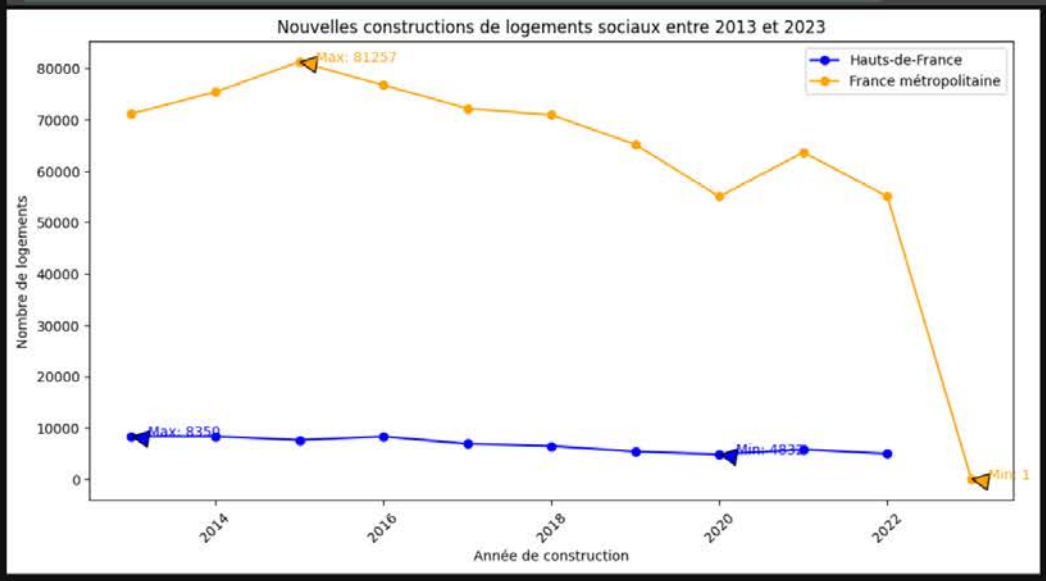
Comparaison des nouvelles constructions sur les 10 dernières années en HDF et FM (2013-2023)

[14]:
```
# Filtrage des dix dernières années
dix_ans = df_log[df_log['Année de construction'] >= 2013]

# Filtrage des données pour les Hauts-de-France
hdf_recent = dix_ans[dix_ans['REG_LIBELLE'] == 'Hauts-de-France']

# Filtrage des données pour la France métropolitaine (excluant les DOM-TOM)
metropole_recent = dix_ans[~dix_ans['REG_LIBELLE'].isin(['Guadeloupe', 'Martinique', 'Guyane', 'La Réunion', 'Mayotte'])]
```

[15]:
```
# Comptage du nombre de logements construits chaque année pour les Hauts-de-France
hdf_constr_dix_ans = hdf_recent.groupby('Année de construction').size()

# Compter le nombre de logements construits chaque année pour la France métropolitaine
metropole_constr_dix_ans = metropole_recent.groupby('Année de construction').size()
```

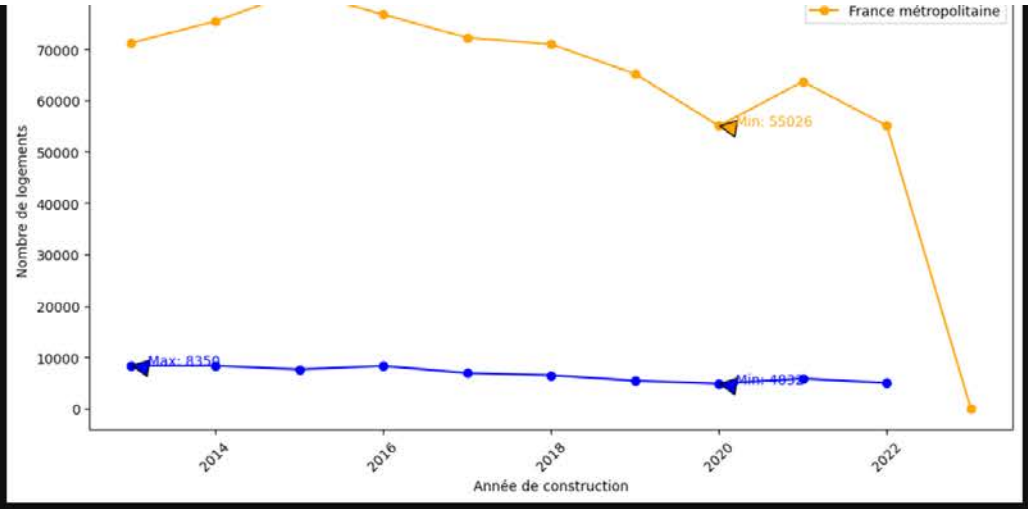
[16]:
```
# Création du graphique pour les dix dernières années
plt.figure(figsize=(12, 6))
```

```
# Ajout des courbes pour les Hauts-de-France et la France métropolitaine
plt.plot(hdf_constr_dix_ans.index, hdf_constr_dix_ans.values, marker='o', color='blue', label='Hauts-de-France')
plt.plot(metropole_constr_dix_ans.index, metropole_constr_dix_ans.values, marker='o', color='orange', label='France métropolitaine')

plt.title('Nouvelles constructions de logements sociaux entre 2013 et 2023')
plt.xlabel('Année de construction')
plt.ylabel('Nombre de logements')
plt.legend()

# Ajout d'une étiquette pour la valeur minimale et maximale des Hauts-de-France
hdf_min_annee = hdf_constr_dix_ans.idxmin()
hdf_max_annee = hdf_constr_dix_ans.idxmax()

plt.annotate(f'Min: {hdf_constr_dix_ans.min():.0f}', xy=(hdf_min_annee, hdf_constr_dix_ans.min()), xytext=(hdf_min_annee + 0.2, hdf_constr_dix_ans.min()
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=10, color='blue')
plt.annotate(f'Max: {hdf_constr_dix_ans.max():.0f}', xy=(hdf_max_annee, hdf_constr_dix_ans.max()), xytext=(hdf_max_annee + 0.2, hdf_constr_dix_ans.max()
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=10, color='blue')

# Ajout d'une étiquette pour la valeur minimale et maximale de la France métropolitaine
metropole_min_annee = metropole_constr_dix_ans.idxmin()
metropole_max_annee = metropole_constr_dix_ans.idxmax()

plt.annotate(f'Min: {metropole_constr_dix_ans.min():.0f}', xy=(metropole_min_annee, metropole_constr_dix_ans.min()), xytext=(metropole_min_annee + 0.2, m
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=10, color='orange')
plt.annotate(f'Max: {metropole_constr_dix_ans.max():.0f}', xy=(metropole_max_annee, metropole_constr_dix_ans.max()), xytext=(metropole_max_annee + 0.2, m
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=10, color='orange')

# Rotation des labels sur l'axe X pour les rendre plus lisibles
plt.xticks(rotation=45)

plt.show()
```



Le minimum de 1 logement construit pour la FM semble aberrant et non représentatif vu que les données s'arrêtent au 01/01/2023. Excluons-le...

[17]:
```
# Création du graphique pour les dix dernières années
plt.figure(figsize=(12, 6))

# Ajout des courbes pour les Hauts-de-France et la France métropolitaine
plt.plot(hdf_constr_dix_ans.index, hdf_constr_dix_ans.values, marker='o', color='blue', label='Hauts-de-France')
plt.plot(metropole_constr_dix_ans.index, metropole_constr_dix_ans.values, marker='o', color='orange', label='France métropolitaine')

plt.title('Nouvelles constructions de logements sociaux entre 2013 et 2023 dans les HDF et en France métropolitaine')
plt.xlabel('Année de construction')
plt.ylabel('Nombre de logements')
plt.legend()

# Ajout d'une étiquette pour la valeur minimale et maximale des Hauts-de-France
hdf_min_annee = hdf_constr_dix_ans.idxmin()
hdf_max_annee = hdf_constr_dix_ans.idxmax()

plt.annotate(f'Min: {hdf_constr_dix_ans.min():.0f}', xy=(hdf_min_annee, hdf_constr_dix_ans.min()), xytext=(hdf_min_annee + 0.2, hdf_constr_dix_ans.min()
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=10, color='blue')
plt.annotate(f'Max: {hdf_constr_dix_ans.max():.0f}', xy=(hdf_max_annee, hdf_constr_dix_ans.max()), xytext=(hdf_max_annee + 0.2, hdf_constr_dix_ans.max()
             arrowprops=dict(facecolor='blue', shrink=0.05), fontsize=10, color='blue')

# Exclusion de l'année 2023 pour la France métropolitaine si elle contient une valeur aberrante
metropole_constr_dix_ans_cleaned = metropole_constr_dix_ans.drop(2023)

# Ajout d'une étiquette pour la nouvelle valeur minimale et maximale de la France métropolitaine
metropole_min_annee = metropole_constr_dix_ans_cleaned.idxmin()
metropole_max_annee = metropole_constr_dix_ans.idxmax()

plt.annotate(f'Min: {metropole_constr_dix_ans_cleaned.min():.0f}', xy=(metropole_min_annee, metropole_constr_dix_ans_cleaned.min()), xytext=(metropole_mi
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=10, color='orange')
plt.annotate(f'Max: {metropole_constr_dix_ans.max():.0f}', xy=(metropole_max_annee, metropole_constr_dix_ans.max()), xytext=(metropole_max_annee + 0.2, m
             arrowprops=dict(facecolor='orange', shrink=0.05), fontsize=10, color='orange')

# Rotation des labels sur l'axe X pour les rendre plus lisibles
plt.xticks(rotation=45)

# Sauvegarde du graphique
plt.savefig('C:/Users/Laetitia_Deken/OneDrive/Bureau/Projet Logement Social 2023/construction_10a_comparison_annotated.png', dpi=300)
plt.show()
```



Ce graphique montre une baisse significative des nouvelles constructions de logements sociaux en France métropolitaine depuis 2017, avec un un sursaut en 2021. La région des Hauts-de-France a connu une stabilité dans les nouvelles constructions. Cette tendance à la baisse en France métropolitaine pourrait refléter des changements dans les politiques publiques ou une réduction des besoins en nouveaux logements sociaux.

Nombre de logements sociaux entre 2013 et 2023 en HDF et FM (logements dans le patrimoine)

La colonne 'Année d'entrée dans le patrimoine' indique quand un logement est intégré au parc locatif et devient disponible pour les locataires sociaux. C'est la raison de ce choix ici.

[28]:
```
# Calcul du total cumulé pour les Hauts-de-France
hdf_total_annuel = df_log[df_log['REG_LIBELLE'] == 'Hauts-de-France'].groupby('Année d'entrée dans le patrimoine').size().cumsum()

# Calcul du total cumulé pour la France métropolitaine (excluant les DOM-TOM)
metropole_total_annuel = df_log[~df_log['REG_LIBELLE'].isin(['Guadeloupe', 'Martinique', 'Guyane', 'La Réunion', 'Mayotte'])].groupby('Année d'entrée dan
```

[29]:
```
# Alignement des deux séries sur le même index
index_commun = hdf_total_annuel.index.union(metropole_total_annuel.index)
hdf_total_annuel = hdf_total_annuel.reindex(index_commun, fill_value=0).ffill()
metropole_total_annuel = metropole_total_annuel.reindex(index_commun, fill_value=0).ffill()

# Création d'un DataFrame pour afficher le total cumulé
df_total_cumule = pd.DataFrame({
    'Année': index_commun,
    'Total cumulé Hauts-de-France': hdf_total_annuel.values,
    'Total cumulé France métropolitaine': metropole_total_annuel.values
})

# Affichage du DataFrame et de ses 10 derniers résultats cumulés
df_total_cumule.tail(10)
```

[29]:

	Année	Total cumulé Hauts-de-France	Total cumulé France métropolitaine
129	2014	518200	4257848
130	2015	526235	4359593
131	2016	536018	4475061
132	2017	551438	4637802
133	2018	560302	4742297
134	2019	565983	4840981
135	2020	577218	4933481
136	2021	587398	5048752
137	2022	593655	5143069
138	2023	0	5143684

0 en total cumulé pour l'année 2023 dans les HDF ? Impossible.

Vérifions pourquoi...

Si nécessaire, on force la valeur de 2023 à être égale à celle de 2022.

[30]:
```
# Vérification si 2023 est présent dans les données originales pour les Hauts-de-France
if 2023 in hdf_total_annuel.index:
    print("2023 est présent dans les données pour les Hauts-de-France.")
    print(hdf_total_annuel.loc[2023])
else:
    print("2023 n'est pas présent dans les données pour les Hauts-de-France.")
```

```
2023 est présent dans les données pour les Hauts-de-France.
0
```

2023 est présent dans les données, mais la valeur est 0 : il n'y a pas eu de nouvelles entrées de logements sociaux pour les Hauts-de-France pour cette année, ce qui est logique, les données s'arrêtant au 01/01/2023.

Il faut donc afficher à la place la dernière valeur connue (celle de 2022) à 2023 pour que 2023 reflète le total cumulé.

[31]:
```
# Alignement des deux séries sur le même index
index_commun = hdf_total_annuel.index.union(metropole_total_annuel.index)

# Si 2023 est manquant ou incorrect, on peut forcer la valeur de 2023 à être égale à celle de 2022
if 2023 not in hdf_total_annuel.index or hdf_total_annuel.loc[2023] == 0:
    hdf_total_annuel.loc[2023] = hdf_total_annuel.loc[2022]
```

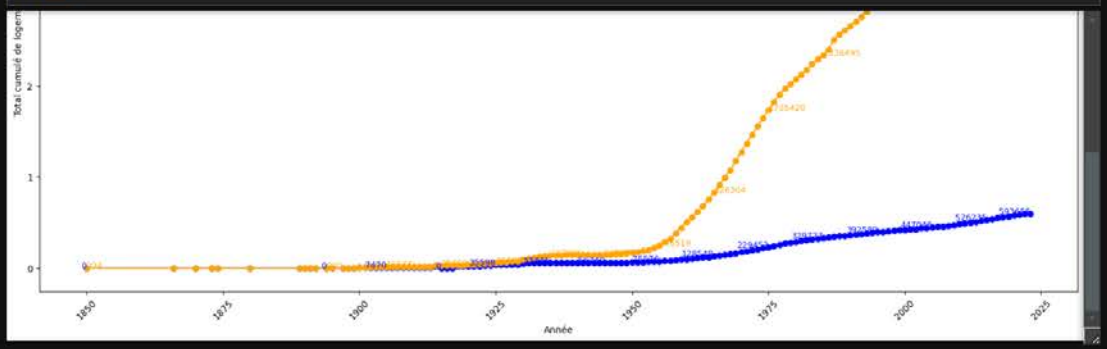
```
# Réindexation et remplissage des valeurs manquantes
hdf_total_annuel = hdf_total_annuel.reindex(index_commun, fill_value=0).ffill()
metropole_total_annuel = metropole_total_annuel.reindex(index_commun, fill_value=0).ffill()

# Création d'un DataFrame pour afficher le total cumulé
df_total_cumule = pd.DataFrame({
    'Année': index_commun,
    'Total cumulé Hauts-de-France': hdf_total_annuel.values,
    'Total cumulé France métropolitaine': metropole_total_annuel.values
})

# Affichage du DataFrame et de ses 10 derniers résultats cumulés
df_total_cumule.tail(10)
```

[31]:

	Année	Total cumulé Hauts-de-France	Total cumulé France métropolitaine
129	2014	518200	4257848
130	2015	526235	4359593
131	2016	536018	4475061
132	2017	551438	4637802
133	2018	560302	4742297
134	2019	565983	4840981
135	2020	577218	4933481
136	2021	587398	5048752
137	2022	593655	5143069
138	2023	593655	5143684

[32]:

```
# Création du graphique pour visualiser l'évolution du total cumulé de logements sociaux
plt.figure(figsize=(20, 10))

# Ajout des courbes pour les Hauts-de-France et la France métropolitaine
plt.plot(hdf_total_annuel.index, hdf_total_annuel.values, marker='o', color='blue', label='Hauts-de-France')
plt.plot(metropole_total_annuel.index, metropole_total_annuel.values, marker='o', color='orange', label='France métropolitaine')

# Ajout des étiquettes pour chaque point (Hauts-de-France) tous les 10 points et pour 2023
for i, (x, y) in enumerate(zip(hdf_total_annuel.index, hdf_total_annuel.values)):
    if i % 10 == 0 or x == 2023:  # Afficher une étiquette tous les 10 points et pour 2023
        plt.text(x, y, f'{y}', fontsize=9, ha='right', color='blue')

# Ajout des étiquettes pour chaque point (France métropolitaine) tous les 10 points et pour 2023
for i, (x, y) in enumerate(zip(metropole_total_annuel.index, metropole_total_annuel.values)):
    if i % 10 == 0 or x == 2023:  # Afficher une étiquette tous les 10 points et pour 2023
        plt.text(x, y, f'{y}', fontsize=9, ha='left', color='orange')

plt.title('Évolution du total cumulé des logements sociaux')
plt.xlabel('Année')
plt.ylabel('Total cumulé de logements sociaux')
plt.legend()

# Rotation des labels sur l'axe X pour les rendre plus lisibles
plt.xticks(rotation=45)

# Sauvegarde du graphique
plt.savefig('C:/Users/Laetitia_Deken/OneDrive/Bureau/Projet logement Social 2023/total_cumul_logements_10a_comparaison.png', dpi=300)
plt.show()
```



On observe une augmentation progressive et continue des logements sociaux en France métropolitaine, avec une nette accélération depuis les années 1950 (après la 2nde GM, donc).

En revanche, la croissance dans les Hauts-de-France est beaucoup plus modérée, restant stable à travers les décennies.

La différence entre les deux courbes indique un écart dans le développement des logements sociaux entre les deux régions, avec la France métropolitaine ayant un rythme de construction beaucoup plus soutenu, dans une logique de reconstruction du pays.

Evolution du nombre de logements sociaux par EPCI (carte longue à charger)

+ 6 cells hidden

Tableau du parc locatif social dans les HDF en janvier 2023

[113]:

```
df_hdf.head(1)
```

[113]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Code Postal
48875	32	Hauts-de-	02	Aisne	200071785	CA Chauny-Tergnier-La	1	pleine	02001	Abbécourt	02300

France Fère propriété

Nombre de logements sociaux

[114]:
```
# Calcul du nombre de logements sociaux par EPCI
logements_sociaux_hdf = df_hdf.groupby('EPCI_LIBELLE').size()
```

Part des résidences en QPV (en %)

[115]:
```
compte_qpv = df_hdf['QPV_LIBELLE'].value_counts()
print(compte_qpv)
```

```
QPV_LIBELLE
Non    383598
Oui    210057
Name: count, dtype: int64
```

[116]:
```
# Calcul du nombre de logements en QPV par EPCI
logements_qpv = df_hdf[df_hdf['QPV_LIBELLE'] == 'Oui'].groupby('EPCI_LIBELLE').size()

# Calcul de la part des résidences en QPV
part_qpv = (logements_qpv / logements_sociaux_hdf) * 100
```

Part des résidences en individuel (en %)

[117]:
```
compte_type = df_hdf['TYPECONST_LIBELLE'].value_counts()
print(compte_type)
```

```
TYPECONST_LIBELLE
collectif           364677
individuel          220326
logement étudiant     8652
Name: count, dtype: int64
```

[118]:
```
# Filtrage des logements individuels
logements_individuels_hdf = df_hdf[df_hdf['TYPECONST_LIBELLE'] == 'individuel'].groupby('EPCI_LIBELLE').size()

# Calcul de la part des logements individuels
part_individuel = (logements_individuels_hdf / logements_sociaux_hdf) * 100
```

Part des résidences en collectif et étudiant (en %)

[119]:
```
# Filtrage des logements collectifs
logements_collectifs_hdf = df_hdf[df_hdf['TYPECONST_LIBELLE'] == 'collectif'].groupby('EPCI_LIBELLE').size()

# Calcul de la part des logements collectifs
part_collectif = (logements_collectifs_hdf / logements_sociaux_hdf) * 100
```

[120]:
```
# Filtrage des logements étudiants
logements_etudiants_hdf = df_hdf[df_hdf['TYPECONST_LIBELLE'] == 'logement étudiant'].groupby('EPCI_LIBELLE').size()

# Calcul de la part des logements étudiants
part_etudiant = (logements_etudiants_hdf / logements_sociaux_hdf) * 100
```

[121]:
```
# Création du DataFrame final
tableau_final = pd.DataFrame({
    'Nombre de logements sociaux': logements_sociaux_hdf,
    'Part de résidences en QPV (%)': part_qpv,
    'Part des logements individuels (%)': part_individuel,
    'Part des logements collectifs (%)': part_collectif,
    'Part des logements étudiants (%)': part_etudiant
})

# Remplacement des NaN par 0 quand il y en a
tableau_final['Part de résidences en QPV (%)'] = tableau_final['Part de résidences en QPV (%)'].fillna(0)
tableau_final['Part des logements individuels (%)'] = tableau_final['Part des logements individuels (%)'].fillna(0)
tableau_final['Part des logements collectifs (%)'] = tableau_final['Part des logements collectifs (%)'].fillna(0)
tableau_final['Part des logements étudiants (%)'] = tableau_final['Part des logements étudiants (%)'].fillna(0)

# Arrondir les colonnes numériques à 2 décimales
tableau_final = tableau_final.round(2)
tableau_final
```

[121]:

EPCI_LIBELLE	Nombre de logements sociaux	Part de résidences en QPV (%)	Part des logements individuels (%)	Part des logements collectifs (%)	Part des logements étudiants (%)
CA Amiens Métropole	26337	43.41	11.82	84.08	4.10
CA Chauny-Tergnier-La Fère	4844	32.20	21.20	78.80	0.00
CA Creil Sud Oise	15309	59.83	5.69	93.68	0.63
CA Douaisis Agglo	19746	29.35	59.41	40.08	0.50
CA Grand Calais Terres et Mers	11848	43.08	23.11	76.89	0.00
CA GrandSoissons Agglomération	7982	47.24	16.34	80.39	3.27
CA Maubeuge Val de Sambre	14927	36.26	39.86	60.14	0.00
CA Valenciennes Métropole	23162	39.16	48.52	48.96	2.52
CA d'Hénin-Carvin	21467	28.77	65.74	34.26	0.00
CA de Béthune-Bruay,Artois-Lys Romane	31661	36.11	72.76	27.24	0.00
CA de Cambrai	4593	26.87	42.89	55.98	1.13
CA de Lens - Liévin	48073	40.99	70.16	29.42	0.42
CA de la Baie de Somme	4435	57.20	26.54	73.46	0.00
CA de la Porte du Hainaut	16630	48.07	69.16	30.84	0.00
CA de la Région de Château-Thierry	3803	31.79	15.01	84.99	0.00
CA de la Région de Compiègne et de la Basse Automne	10172	38.31	9.41	87.95	2.64

CA des Deux Baies en Montreuillois	4255	9.24	39.32	60.68	0.00
CA du Beauvaisis	12213	47.16	17.92	78.49	3.59
CA du Boulonnais	15496	37.68	24.08	74.83	1.09
CA du Caudrésis et du Catésis	2251	8.62	38.65	61.35	0.00
CA du Pays de Laon	5974	40.54	24.56	73.18	2.26
CA du Pays de Saint-Omer	6360	34.59	33.22	65.99	0.79
CA du Saint-Quentinois	8743	50.93	24.33	75.02	0.65
CC Avre Luce Noye	436	0.00	49.08	50.92	0.00
CC Cœur d'Ostrevent	8866	46.41	83.13	16.87	0.00
CC Cœur de l'Avesnois	1306	0.00	31.62	68.38	0.00
CC Flandre Lys	2200	0.00	71.14	28.86	0.00
CC Interrégionale Aumale - Blangy-sur-Bresle	8	0.00	0.00	100.00	0.00
CC Nièvre et Somme	886	0.00	82.05	17.95	0.00
CC Osartis Marquion	1128	0.00	71.01	28.99	0.00
CC Pays d'Opale	790	0.00	76.58	23.42	0.00
CC Picardie des Châteaux	585	0.00	31.11	66.15	2.74
CC Ponthieu-Marquenterre	622	0.00	81.35	18.65	0.00
CC Pévèle-Carembault	4418	13.06	62.04	37.96	0.00
CC Retz-en-Valois	2114	24.50	16.46	83.54	0.00
CC Senlis Sud Oise	2062	0.00	4.46	95.54	0.00
CC Somme Sud-Ouest	853	0.00	79.95	20.05	0.00
CC Terre de Picardie	629	0.00	69.79	30.21	0.00
CC Thelloise	3261	0.00	31.62	68.38	0.00
CC Thiérache Sambre et Oise	719	0.00	37.69	62.31	0.00
CC de Desvres-Samer	778	0.00	68.51	31.49	0.00
CC de Flandre Intérieure	5486	7.69	56.56	43.44	0.00
CC de l'Aire Cantilienne	2459	0.00	8.46	91.54	0.00
CC de l'Est de la Somme	979	0.00	76.92	23.08	0.00
CC de l'Oise Picarde	959	0.00	58.81	41.19	0.00
CC de la Champagne Picarde	575	0.00	48.52	51.48	0.00
CC de la Haute-Somme	1291	0.00	31.68	68.32	0.00
CC de la Picardie Verte	1537	0.00	54.39	45.61	0.00
CC de la Plaine d'Estrées	590	0.00	38.64	61.36	0.00
CC de la Région d'Audruicq	959	0.00	80.92	19.08	0.00
CC de la Terre des Deux Caps	1193	14.59	82.73	17.27	0.00
CC de la Thiérache du Centre	1439	0.00	42.95	54.83	2.22
CC des Campagnes de l'Artois	396	0.00	77.78	22.22	0.00
CC des Deux Vallées	1804	0.00	13.69	86.31	0.00
CC des Hauts de Flandre	2064	0.00	72.04	27.96	0.00
CC des Lisières de l'Oise	556	0.00	25.72	74.28	0.00
CC des Pays d'Oise et d'Halatte	2681	20.33	17.94	82.06	0.00
CC des Portes de la Thiérache	231	0.00	29.44	70.56	0.00
CC des Sablons	2413	39.87	23.75	76.25	0.00
CC des Sept Vallées	1213	0.00	78.40	21.60	0.00
CC des Trois Rivières	1277	40.09	34.46	65.54	0.00
CC des Villes Sœurs	718	0.00	40.39	59.61	0.00
CC du Canton d'Oulchy-le-Château	79	0.00	51.90	48.10	0.00
CC du Canton de Charly-sur-Marne	346	0.00	12.43	87.57	0.00
CC du Chemin des Dames	84	0.00	85.71	14.29	0.00
CC du Clermontois	2921	17.12	14.38	85.62	0.00
CC du Grand Roye	1687	0.00	43.86	56.14	0.00
CC du Haut Pays du Montreuillois	309	0.00	77.02	22.98	0.00
CC du Liancourtois	1616	19.25	9.28	90.72	0.00
CC du Pays Noyonnais	2620	33.13	24.20	75.80	0.00
CC du Pays Solesmois	250	0.00	70.00	30.00	0.00
CC du Pays de Bray	864	0.00	46.88	53.12	0.00
CC du Pays de Lumbres	564	0.00	84.40	15.60	0.00
CC du Pays de Mormal	1867	21.00	53.24	46.76	0.00
CC du Pays de Valois	2523	29.21	12.60	87.40	0.00
CC du Pays de la Serre	500	0.00	47.40	52.60	0.00
CC du Pays des Sources	452	0.00	45.35	54.65	0.00
CC du Pays du Coquelicot	791	0.00	41.34	58.66	0.00

CC du Pays du Vermandois	976	0.00	39.34	60.66	0.00
CC du Plateau Picard	1540	0.00	37.01	62.99	0.00
CC du Sud Avesnois	2591	46.35	38.98	61.02	0.00
CC du Sud-Artois	870	0.00	73.33	26.67	0.00
CC du Ternois	1209	0.00	59.88	40.12	0.00
CC du Territoire Nord Picardie	997	0.00	47.74	52.26	0.00
CC du Val de Somme	860	0.00	66.28	33.72	0.00
CC du Val de l'Aisne	515	0.00	50.87	49.13	0.00
CC du Val de l'Oise	350	0.00	76.29	23.71	0.00
CC du Vexin-Thelle	410	0.00	50.00	50.00	0.00
CC du Vimeu	871	0.00	55.57	44.43	0.00
CU d'Arras	12520	45.14	18.46	80.34	1.20
CU de Dunkerque	29431	31.97	26.53	73.11	0.36
Métropole Européenne de Lille	140280	40.21	19.73	76.81	3.46

Répartition en QPV (Quartiers Prioritaires de la Politique de la Ville) :

- Certaines agglomérations comme CA Creil Sud Oise (59,83%) et CA Amiens Métropole (43,41%) ont une proportion significative de logements situés dans des QPV, ce qui reflète une concentration plus élevée de logements dans des zones prioritaires.
- En revanche, plusieurs communautés de communes (comme CC Flandre Lys et CC Somme Sud-Ouest) n'ont aucun logement social situé dans un QPV.

Type de logement :

- Logements collectifs dominent dans la plupart des EPCI. Par exemple, CA Creil Sud Oise a 93,68% de logements collectifs, ce qui montre une urbanisation dense.
- En revanche, des agglomérations comme CA d'Hénin-Carvin et CA de Béthune-Bruay, Artois-Lys Romane ont une forte proportion de logements individuels (65,74% et 72,76% respectivement), indiquant une préférence pour les logements moins denses.

Logements étudiants :

- La part des logements étudiants est généralement faible, sauf dans certaines agglomérations comme CA Amiens Métropole (4,10%) et CA GrandSoissons Agglomération (3,27%). La faible proportion dans la majorité des autres EPCI reflète peut-être une moindre présence d'établissements d'enseignement supérieur.

- Comparaison des EPCI :

- La Métropole Européenne de Lille possède le plus grand nombre de logements sociaux (140280), ce qui est logique compte tenu de sa taille et de son importance économique dans la région.

- En contraste, des communautés comme CC Cœur d'Ostrevent et CC Nièvre et Somme montrent une forte préférence pour les logements individuels (83,13% et 82,05% respectivement), ce qui peut refléter une moindre densité de population ou une planification urbaine différente.

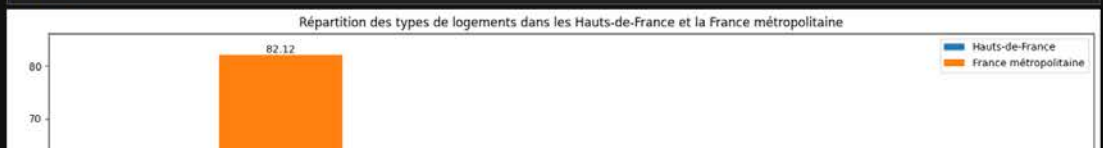
Caractéristiques du parc social

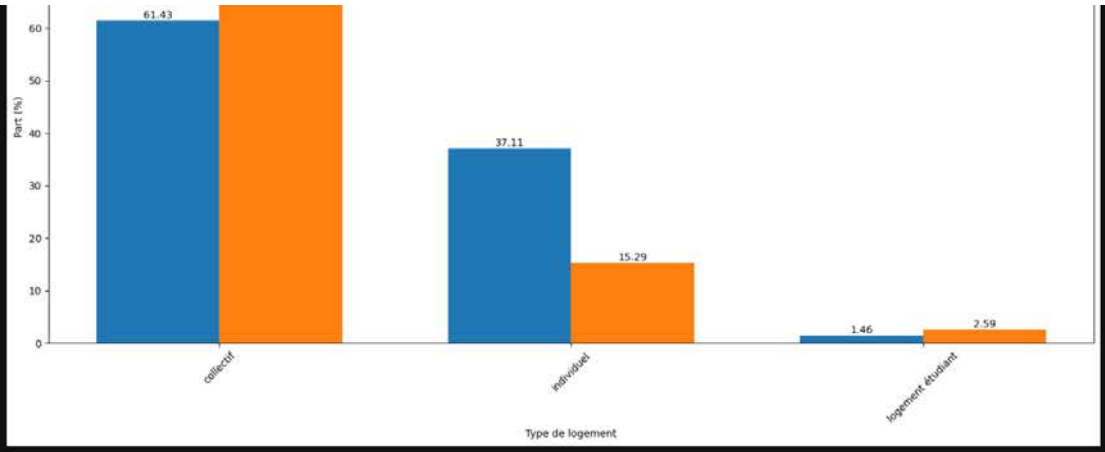
Part des logements individuels, collectifs et étudiants en janvier 2023

[128]:
```
# Calcul des parts pour les Hauts-de-France
hdf_donnees = df_log[df_log['REG_LIBELLE'] == 'Hauts-de-France']
hdf_parts = hdf_donnees['TYPECONST_LIBELLE'].value_counts(normalize=True) * 100

# Calcul des parts pour la France métropolitaine (excluant les DOM-TOM)
metropole_donnees = df_log[~df_log['REG_LIBELLE'].isin(['Guadeloupe', 'Martinique', 'Guyane', 'La Réunion', 'Mayotte'])]
metropole_parts = metropole_donnees['TYPECONST_LIBELLE'].value_counts(normalize=True) * 100
```

[129]:
```
# Combinaison des données pour les Hauts-de-France et la France métropolitaine
parts_combine = pd.DataFrame({
    'Hauts-de-France': hdf_parts,
    'France métropolitaine': metropole_parts
})

# Assurez-vous que les colonnes sont alignées, remplissez les NaN avec 0
parts_combine = parts_combine.fillna(0)
```

[130]:
```
plt.figure(figsize=(15, 8))

indices = np.arange(len(parts_combine))
bar_width = 0.35
bars_hdf = plt.bar(indices, parts_combine['Hauts-de-France'], bar_width, label='Hauts-de-France')
bars_metropole = plt.bar(indices + bar_width, parts_combine['France métropolitaine'], bar_width, label='France métropolitaine')

# Ajouter les étiquettes sur les barres
for bar in bars_hdf:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2), ha='center', va='bottom')

for bar in bars_metropole:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2), ha='center', va='bottom')

plt.xlabel('Type de logement')
plt.ylabel('Part (%)')
plt.title('Répartition des types de logements dans les Hauts-de-France et la France métropolitaine')

plt.xticks(indices + bar_width / 2, parts_combine.index, rotation=45)
plt.legend()

plt.tight_layout()
plt.show()
```





- En France métropolitaine, les logements collectifs représentent la grande majorité du parc de logements sociaux, avec 82,1 % des logements.
- Dans les Hauts-de-France, cette part est légèrement moins importante mais reste majoritaire, avec 61,4 %. Cette prévalence soit plus marquée au niveau national.

La région Hauts-de-France montre une part plus importante de logements individuels (37 %) par rapport à la moyenne nationale (15 %). Cela reflète une tradition ou une politique locale plus orientée vers des logements individuels dans cette région, en comparaison avec le reste de la France.

Les logements étudiants sont présents dans une proportion beaucoup plus faible en général. Elle représente 1,5 % dans les Hauts-de-France comparé à la France métropolitaine (2,6%). Cela pourrait indiquer une densité plus faible d'institutions universitaires ou une politique d'offre de logements étudiants différente dans cette région.

Répartition des logements sociaux selon le nombre de pièces

Les mises en services

En partant, bien évidemment, du DataFrame df_hdf dispo précédemment...

Tableau récap des mises en service en 2022 dans les HDF

Nombre de logements pour 2022 et 2023 dans les HDF

Mises en service

[125]:
```
# Calcul des mises en service pour 2023
mises_en_service_2023 = df_hdf[df_hdf['Année d'entrée dans le patrimoine'] == 2022].groupby('DEP_LIBELLE').size() # état en janvier 2023
mises_en_service_2023
```

[125]:
```
DEP_LIBELLE
Aisne            384
Nord            3154
Oise             847
Pas-de-Calais   1456
Somme            416
dtype: int64
```

Acquisitions

```
Colonne "Origine de l'entrée dans le patrimoine":

- 1 = Construction par l'organisme
- 2 = Acquisition avec travaux
- 3 = Acquisition sans travaux
- 4 = Acquisition en Vefa (Acte de vente d'un logement en l'état futur d'achèvement)
- 0 = Non renseigné
- NC = Non conforme
```

[126]:
```
# Conversion de la colonne en entier, en ignorant les erreurs (par exemple pour 'NC')
df_hdf['Origine de l'entrée dans le patrimoine'] = pd.to_numeric(df_hdf['Origine de l'entrée dans le patrimoine'], errors='coerce')

# Comptage des ventes par département
acquisitions = df_hdf[(df_hdf['Origine de l'entrée dans le patrimoine'].isin([2, 3, 4])) &
                      (df_hdf['Année d'entrée dans le patrimoine'] == 2022)].groupby('DEP_LIBELLE').size()
```

```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\201938491.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Origine de l'entrée dans le patrimoine'] = pd.to_numeric(df_hdf['Origine de l'entrée dans le patrimoine'], errors='coerce')
```

[127]:
```
# Création du tableau final
tableau_mouvements = pd.DataFrame({
    'Total logements 2023': dep_libelle_2023,
    'Total logements 2022': dep_libelle_2022,
    'Mises en service': mises_en_service_2023,
    'Acquisitions': acquisitions,
}).fillna(0)

tableau_mouvements
```

[127]:

DEP_LIBELLE	Total logements 2023	Total logements 2022	Mises en service	Acquisitions
Aisne	384	254	384	47
Nord	3154	7322	3154	1253
Oise	847	738	847	679
Pas-de-Calais	1456	1395	1456	783
Somme	416	471	416	136

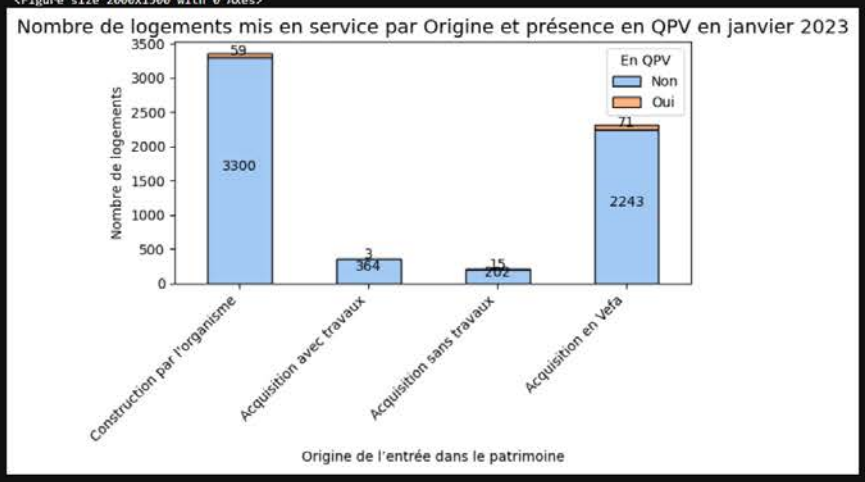
Logements mis en service en 2022 hors et en QPV

[147]:
```
# Filtrage des données pour 2023 et 2022
data_2023 = df_hdf[df_hdf['Année d'entrée dans le patrimoine'] == 2022] # Total en janvier 2023
```

[148]:
```
# Segmentation par Origine de l'entrée dans le patrimoine et QPV_LIBELLE
logements_groupe = data_2023.groupby(['Origine de l'entrée dans le patrimoine', 'QPV_LIBELLE']).size().unstack()
```

[153]:
```
# Dictionnaire pour remplacer les chiffres par les termes appropriés
origine_etiq = {
    1: "Construction par l'organisme",
    2: "Acquisition avec travaux",
    3: "Acquisition sans travaux",
    4: "Acquisition en Vefa"
}

# Création du graphique
plt.figure(figsize=(20, 15))
colors = sns.color_palette("pastel")
ax = logements_groupe.plot(kind='bar', stacked=True, color=colors, edgecolor='black')

# Remplacement des chiffres 1,2,3 et 4 par la variable origine_etiq dans l'axe X
ax.set_xticklabels([origine_etiq[i] for i in logements_groupe.index])

# Ajouter des étiquettes sur chaque barre
for container in ax.containers:
    ax.bar_label(container, label_type='center', fmt='%d', fontsize=10, color='black', padding=3)

plt.title("Nombre de logements mis en service par Origine et présence en QPV en janvier 2023", fontsize=14)
plt.xlabel("Origine de l'entrée dans le patrimoine", fontsize=10)
plt.ylabel("Nombre de logements", fontsize=10)
plt.legend(title="En QPV", labels=['Non', 'Oui'], fontsize=10)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.yticks(fontsize=10)
plt.grid(False)
plt.tight_layout()
plt.show()
```

```
<Figure size 2000x1500 with 0 Axes>
```



Les constructions par l'organisme sont la méthode prédominante pour les mises en service avec 3 300 logements, dont 59 en QPV. Cela montre une forte tendance à la construction directe par les organismes.

"Acquisition en VEFA" (Vente en l'état futur d'achèvement) représente une autre méthode importante avec 2 243 logements, dont 71 en QPV. Cela indique une autre voie significative d'entrée de logements dans le parc social.

Les acquisitions avec et sans travaux représentent une part beaucoup plus faible des mises en service, avec respectivement 364 et 202 logements, dont une toute petite proportion est en QPV.

Ici, le graphique met en évidence que la construction par l'organisme et l'acquisition en VEFA sont les deux principales méthodes de mise en service des logements sociaux, avec une part plutôt modeste de logements situés en QPV.

L'ancienneté et l'état énergétique du parc social

Tableau récap des logements sociaux selon leur ancienneté en janvier 2023 (première mise en location)

[169]:
```
# Définition de l'année référence
annee_ref = 2023

# Définition des tranches d'âge
bins = [0, 4, 9, 19, 39, 59, float('inf')]
labels = ['Moins de 5 ans', '5 à 9 ans', '10 à 19 ans', '20 à 39 ans', '40 à 59 ans', 'Plus de 60 ans']

# Calcul de l'âge des logements et création de la colonne "Tranche d'âge"
df_hdf['Âge du logement'] = annee_ref - df_hdf['Année de première mise en location']
df_hdf['Tranche d\'âge'] = pd.cut(df_hdf['Âge du logement'], bins=bins, labels=labels, include_lowest=True)

# Groupement des données par département et tranche d'âge, puis compter le nombre de logements
tableau_recap = df_hdf.groupby(['DEP_LIBELLE', 'Tranche d\'âge']).size().unstack().fillna(0)

# Renommage des colonnes pour qu'elles correspondent à la tranche d'âge voulue
tableau_recap.columns = ['Logements âgés de moins de 5 ans',
                         'Logements âgés de 5 à 9 ans',
                         'Logements âgés de 10 à 19 ans',
                         'Logements âgés de 20 à 39 ans',
                         'Logements âgés de 40 à 59 ans',
                         'Logements âgés de plus de 60 ans']

tableau_recap
```

```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\4029125283.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Âge du logement'] = annee_ref - df_hdf['Année de première mise en location']
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\4029125283.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Tranche d\'âge'] = pd.cut(df_hdf['Âge du logement'], bins=bins, labels=labels, include_lowest=True)
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\4029125283.py:13: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  tableau_recap = df_hdf.groupby(['DEP_LIBELLE', 'Tranche d\'âge']).size().unstack().fillna(0)
```

[169]:

DEP_LIBELLE	Logements âgés de moins de 5 ans	Logements âgés de 5 à 9 ans	Logements âgés de 10 à 19 ans	Logements âgés de 20 à 39 ans	Logements âgés de 40 à 59 ans	Logements âgés de plus de 60 ans
Aisne	1011	1422	4186	9791	20531	4195
Nord	15106	22703	35544	53467	101102	51510
Oise	3157	4994	7666	16954	29301	6890
Pas-de-Calais	5000	9400	18069	31690	46448	51118
Somme	2142	3368	6130	8705	17305	4750

[177]:
```
# Définition de l'année référence
annee_ref = 2023

# Définition des tranches d'âge
bins = [0, 4, 9, 19, 39, 59, float('inf')]
labels = ['Moins de 5 ans', '5 à 9 ans', '10 à 19 ans', '20 à 39 ans', '40 à 59 ans', 'Plus de 60 ans']

# Calcul de l'âge des logements et création de la colonne "Tranche d'âge"
df_hdf['Âge du logement'] = annee_ref - df_hdf['Année de première mise en location']
df_hdf['Tranche d\'âge'] = pd.cut(df_hdf['Âge du logement'], bins=bins, labels=labels, include_lowest=True)

# Groupement des données par département et tranche d'âge, puis compter le nombre de logements
tableau_recap = df_hdf.groupby(['DEP_LIBELLE', 'Tranche d\'âge']).size().unstack().fillna(0)

# Calcul du total des logements par département
total_logements_dep = tableau_recap.sum(axis=1)

# Calcul des pourcentages
tableau_recap_pct = tableau_recap.div(total_logements_dep, axis=0) * 100

# Calcul de la moyenne des pourcentages pour chaque tranche d'âge
moyenne_hdf = tableau_recap_pct.mean()

# Ajout au DataFrame sous la forme d'une nouvelle ligne
tableau_recap_pct.loc['Hauts-de-France'] = moyenne_hdf
tableau_recap_pct = tableau_recap_pct.round(2)
tableau_recap_pct
```

```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\3106130837.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Âge du logement'] = annee_ref - df_hdf['Année de première mise en location']
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\3106130837.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Tranche d\'âge'] = pd.cut(df_hdf['Âge du logement'], bins=bins, labels=labels, include_lowest=True)
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_23872\3106130837.py:13: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  tableau_recap = df_hdf.groupby(['DEP_LIBELLE', 'Tranche d\'âge']).size().unstack().fillna(0)
```

[177]:

DEP_LIBELLE	Tranche d'âge	Moins de 5 ans	5 à 9 ans	10 à 19 ans	20 à 39 ans	40 à 59 ans	Plus de 60 ans
Aisne		2.46	3.46	10.18	23.80	49.91	10.20
Nord		5.41	8.12	12.72	19.13	36.18	18.43
Oise		4.58	7.24	11.12	24.58	42.49	9.99
Pas-de-Calais		3.09	5.81	11.17	19.59	28.72	31.61
Somme		5.05	7.94	14.46	20.53	40.81	11.20
Hauts-de-France		4.12	6.52	11.93	21.53	39.62	16.29

Le tableau montre que la majorité des logements sociaux dans les Hauts-de-France sont anciens, avec plus de 55 % ayant plus de 40 ans. Le Pas-de-Calais a la plus grande proportion de logements très anciens (31.61 % ont plus de 60 ans). Les logements récents (moins de 10 ans) sont peu nombreux, ce qui souligne un renouvellement faible du parc de logements sociaux dans la région.

Répartition des logements sociaux selon leur consommation d'énergie

[180]:
```
df_hdf.head(1)
```

[180]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Code Postal
48875	32	Hauts-de-France	02	Aisne	200071785	CA Chauny-Tergnier-La	1	pleine propriété	02001	Abbécourt	02300

Fère

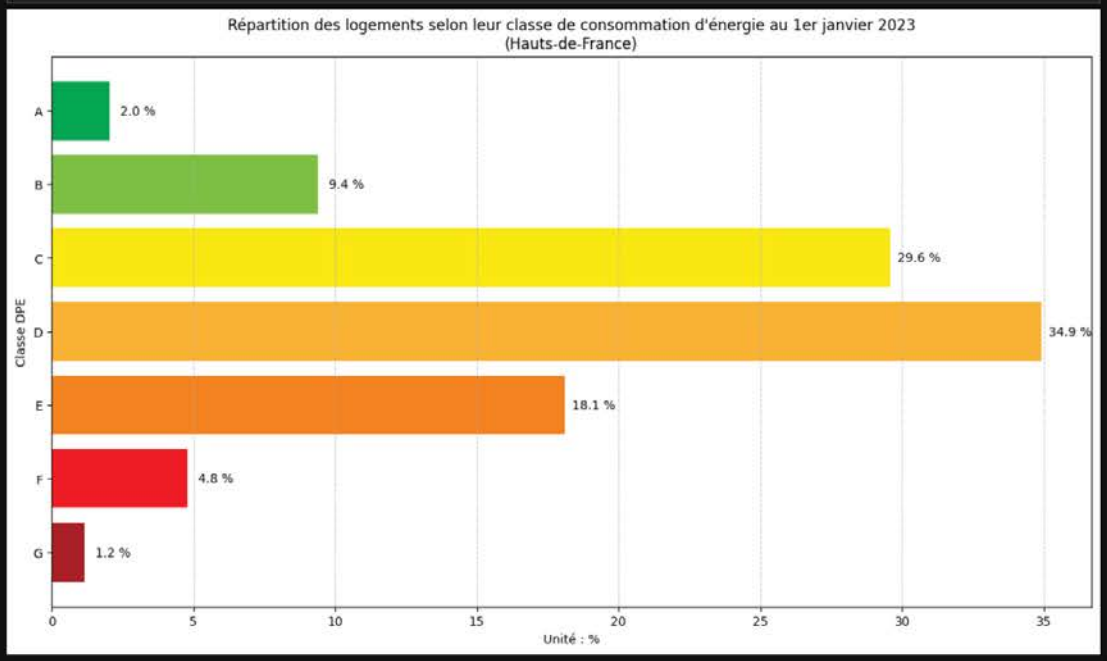
[186]:
```
compte_dpe = df_hdf['Etiquette DPE Energie'].value_counts().sort_index()
compte_dpe
```

[186]:
```
Etiquette DPE Energie
      113284
A       9804
B      45149
C     142111
D     167723
E      86964
F      23057
G       5563
Name: count, dtype: int64
```

```
# Comptage du nombre de logements par classe DPE •••
```

```
# Calcul des pourcentages •••
```

```
# Filtrage des données pour exclure les valeurs sans étiquette DPE •••
```



La majorité des logements se trouvent dans les classes D (34.9 %) et C (29.6 %), indiquant une efficacité énergétique modérée. Les classes les plus performantes (A et B) représentent une petite fraction, tandis que les classes les moins performantes (F et G) sont également minoritaires, avec une part combinée de 6 %. Ce profil suggère que la plupart des logements nécessitent des améliorations pour atteindre des standards énergétiques plus élevés.

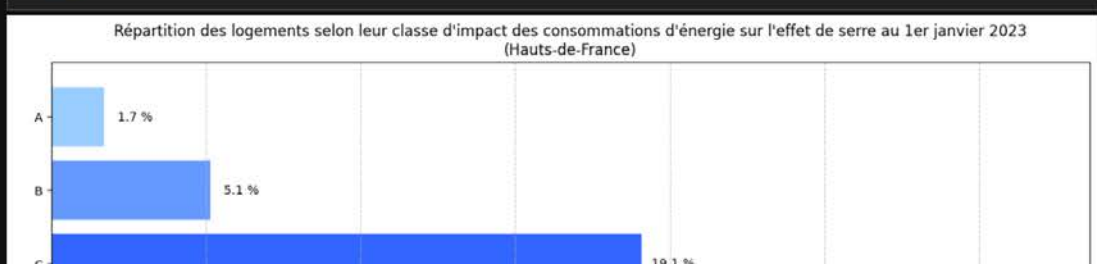
Répartition des logements sociaux selon leur consommation d'énergie sur l'effet de serre

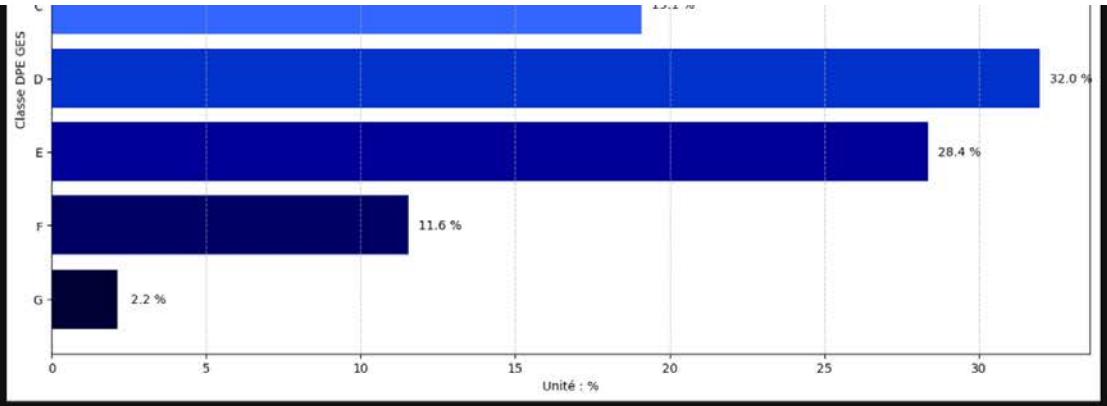
[22]:
```
# Filtrage des données pour exclure les valeurs sans étiquette DPE GES
df_dpe_ges_filtre = df_hdf[df_hdf['Etiquette DPE GES'].isin(['A', 'B', 'C', 'D', 'E', 'F', 'G'])]
```

[23]:
```
# Comptage du nombre de logements par étiquette DPE GES
compte_dpe_ges = df_dpe_ges_filtre['Etiquette DPE GES'].value_counts().sort_index()
```

[24]:
```
# Calcul des pourcentages
total_logements_ges = compte_dpe_ges.sum()
pourcentages_dpe_ges = (compte_dpe_ges / total_logements_ges) * 100
```

[27]:
```
# Création du graphique
plt.figure(figsize=(15, 8))
barres = plt.barh(compte_dpe_ges.index, pourcentages_dpe_ges, color=['#99ccff', '#6699ff', '#3366ff', '#0033cc', '#000099', '#000066', '#000033'])

# Ajout des étiquettes sur chaque barre
for barre in barres:
    width = barre.get_width()
    plt.text(width + 1, barre.get_y() + barre.get_height()/2,
             f'{width:.1f} %', ha='center', va='center')

# Ajout d'un titre et des labels
plt.title("Répartition des logements selon leur classe d'impact des consommations d'énergie sur l'effet de serre au 1er janvier 2023\n(Hauts-de-France)")
plt.xlabel('Unité : %')
plt.ylabel('Classe DPE GES')

plt.grid(True, axis='x', linestyle='--', alpha=0.7)
plt.gca().invert_yaxis()
plt.show()
```





Ce graphique illustre la répartition des logements sociaux dans les Hauts-de-France en fonction de leur classe GES (gaz à effet de serre). Les classes C (32.0 %) et D (28.4 %) dominent, ce qui indique que la majorité des logements se situe dans des niveaux intermédiaires d'émission de gaz à effet de serre. Les logements classés A et B, représentant les niveaux les plus faibles d'émissions, sont minoritaires avec respectivement 1.7 % et 5.1 %. À l'inverse, les logements avec des émissions élevées (classes F et G) constituent une part non négligeable, soulignant l'importance de l'amélioration des performances énergétiques dans cette région.

Répartition des logements par catégorie de financement

[29]:
```
# Hauts-de-France
tableau_hdf = df_hdf.groupby('Catégorie financement CUS').size().reset_index(name='Nombre de logements')
tableau_hdf['Pourcentage'] = (tableau_hdf['Nombre de logements'] / tableau_hdf['Nombre de logements'].sum()) * 100

# France métropolitaine
tableau_metropole = df_metropole.groupby('Catégorie financement CUS').size().reset_index(name='Nombre de logements')
tableau_metropole['Pourcentage'] = (tableau_metropole['Nombre de logements'] / tableau_metropole['Nombre de logements'].sum()) * 100

# Pourcentages à 2 décimales
tableau_hdf['Pourcentage'] = tableau_hdf['Pourcentage'].round(2)
tableau_metropole['Pourcentage'] = tableau_metropole['Pourcentage'].round(2)
```

[34]:
```
# Fusion des deux tableaux sur la colonne 'Catégorie financement CUS'
tableau_comparatif = pd.merge(tableau_hdf[['Catégorie financement CUS', 'Pourcentage']],
                              tableau_metropole[['Catégorie financement CUS', 'Pourcentage']],
                              on='Catégorie financement CUS',
                              suffixes=('_Hauts-de-France', '_France métropolitaine'))

tableau_comparatif
```

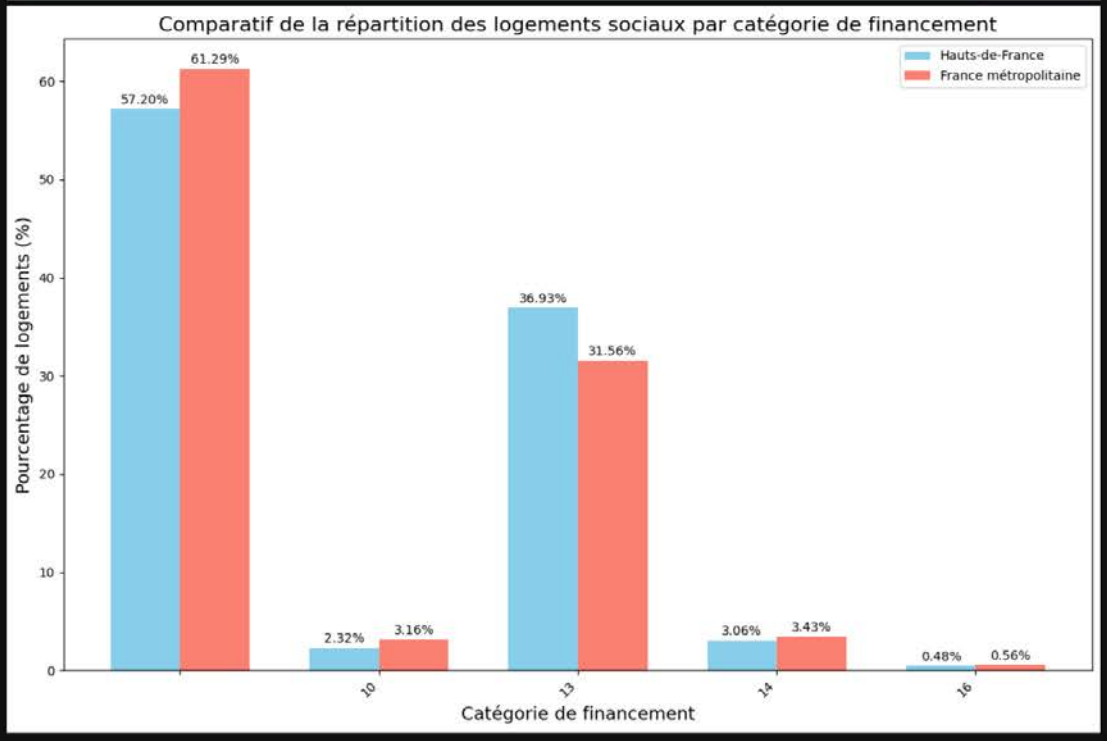
[34]:

	Catégorie financement CUS	Pourcentage_Hauts-de-France	Pourcentage_France métropolitaine
0		57.20	61.29
1	10	2.32	3.16
2	13	36.93	31.56
3	14	3.06	3.43
4	16	0.48	0.56

- 10 = 10.PLA d'intégration - LLTS dans les DOM
- 11 = 11.PLA LM/PLATS/PLAI - LLS dans les DOM
- 12 = 12.PLA social / PLA ordinaire
- 13 = 13.PLUS
- 14 = 14.PLS/PPLS/PLA CFF
- 15 = 15.PAP locatif
- 16 = 16.PLI
- 17 = 17.PCL (conventionné ou non) / PCLS
- 49 = 49.Autre financement à partir de 1977 (1983 dans les DOM)
- 50 = 50.HBM
- 51 = 51.PLR/PSR
- 52 = 52.HLM/O
- 53 = 53.ILM
- 54 = 54.ILN
- 55 = 55.Prêts spéciaux du CFF
- 99 = 99.Autre financement avant 1977 (1983 dans les DOM)
- 0 = Non renseigné
- NC = non conforme

[36]:
```
fig, ax = plt.subplots(figsize=(12, 8))

# Barres
bar_width = 0.35
index = range(len(tableau_comparatif))

bar_hdf = ax.bar(index, tableau_comparatif['Pourcentage_Hauts-de-France'], bar_width, label='Hauts-de-France', color='skyblue')
bar_metropole = ax.bar([i + bar_width for i in index], tableau_comparatif['Pourcentage_France métropolitaine'], bar_width, label='France métropolitaine',

# Ajout des étiquettes de pourcentage sur chaque barre
for i in index:
    ax.text(i, tableau_comparatif['Pourcentage_Hauts-de-France'][i] + 0.5,
            f"{tableau_comparatif['Pourcentage_Hauts-de-France'][i]:.2f}%", ha='center', fontsize=10)
    ax.text(i + bar_width, tableau_comparatif['Pourcentage_France métropolitaine'][i] + 0.5,
            f"{tableau_comparatif['Pourcentage_France métropolitaine'][i]:.2f}%", ha='center', fontsize=10)

ax.set_title("Comparatif de la répartition des logements sociaux par catégorie de financement", fontsize=16)
ax.set_xlabel("Catégorie de financement", fontsize=14)
ax.set_ylabel("Pourcentage de logements (%)", fontsize=14)
ax.set_xticks([i + bar_width / 2 for i in index])
ax.set_xticklabels(tableau_comparatif['Catégorie financement CUS'], rotation=45, ha='right')
ax.legend()
```

```
plt.tight_layout()
plt.show()
```



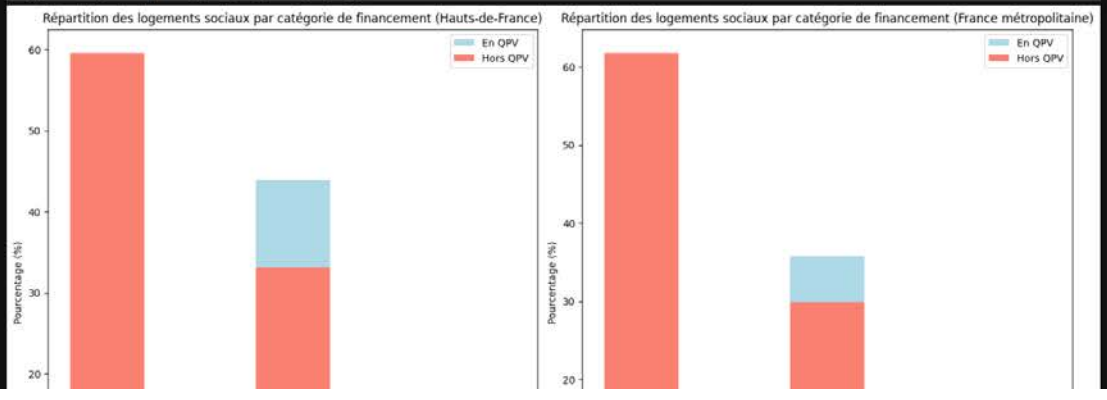
Analyse des logements en Quartiers Prioritaires de la Politique de la Ville (QPV)

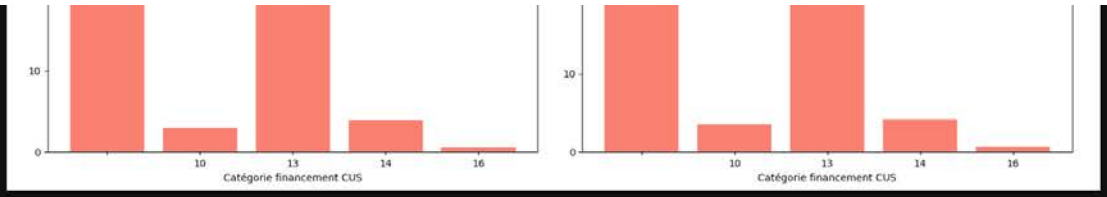
[37]:
```
# Logements en QPV dans les Hauts-de-France
logements_qpv_hdf = df_hdf[df_hdf['QPV_LIBELLE'] == 'Oui']

# Logements hors QPV dans les Hauts-de-France
logements_hors_qpv_hdf = df_hdf[df_hdf['QPV_LIBELLE'] == 'Non']

# Logements en QPV en France métropolitaine
logements_qpv_metropole = df_metropole[df_metropole['QPV_LIBELLE'] == 'Oui']

# Logements hors QPV en France métropolitaine
logements_hors_qpv_metropole = df_metropole[df_metropole['QPV_LIBELLE'] == 'Non']
```

[38]:
```
# Répartition par catégorie de financement dans les QPV en Hauts-de-France
financement_qpv_hdf = logements_qpv_hdf.groupby('Catégorie financement CUS').size()

# Répartition par catégorie de financement hors QPV en Hauts-de-France
financement_hors_qpv_hdf = logements_hors_qpv_hdf.groupby('Catégorie financement CUS').size()

# Répartition par catégorie de financement dans les QPV en France métropolitaine
financement_qpv_metropole = logements_qpv_metropole.groupby('Catégorie financement CUS').size()

# Répartition par catégorie de financement hors QPV en France métropolitaine
financement_hors_qpv_metropole = logements_hors_qpv_metropole.groupby('Catégorie financement CUS').size()
```

[39]:
```
# Pourcentages des catégories de financement pour les logements en QPV (Hauts-de-France)
pourcentage_qpv_hdf = (financement_qpv_hdf / financement_qpv_hdf.sum()) * 100

# Pourcentages des catégories de financement pour les logements hors QPV (Hauts-de-France)
pourcentage_hors_qpv_hdf = (financement_hors_qpv_hdf / financement_hors_qpv_hdf.sum()) * 100

# Pourcentages des catégories de financement pour les logements en QPV (France métropolitaine)
pourcentage_qpv_metropole = (financement_qpv_metropole / financement_qpv_metropole.sum()) * 100

# Pourcentages des catégories de financement pour les logements hors QPV (France métropolitaine)
pourcentage_hors_qpv_metropole = (financement_hors_qpv_metropole / financement_hors_qpv_metropole.sum()) * 100
```

```
fig, axs = plt.subplots(1, 2, figsize=(15, 8))
```



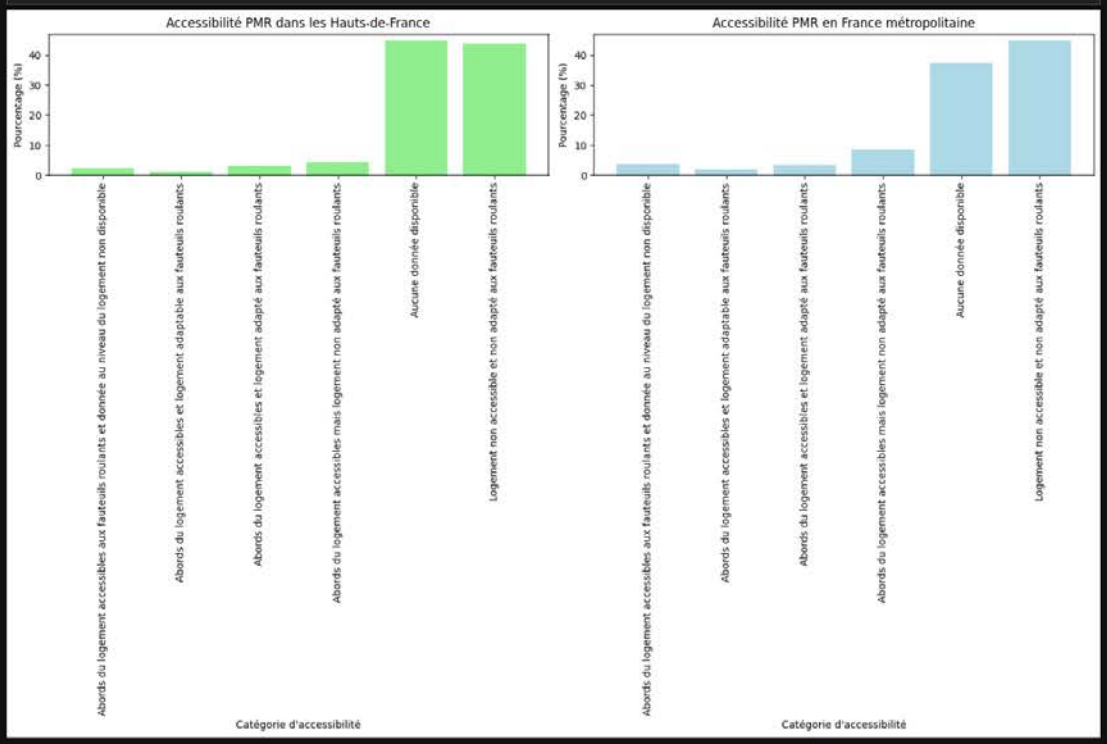


Analyse de l'accessibilité pour les personnes à mobilité réduite (PMR)

[44]:
```
# Analyse de l'accessibilité pour les PMR dans les Hauts-de-France
accessibilite_hdf = df_hdf.groupby('PMR_LIBELLE').size()

# Analyse de l'accessibilité pour les PMR en France métropolitaine
accessibilite_metropole = df_metropole.groupby('PMR_LIBELLE').size()
```

[45]:
```
# Pourcentages d'accessibilité PMR pour les Hauts-de-France
pourcentage_accessibilite_hdf = (accessibilite_hdf / accessibilite_hdf.sum()) * 100

# Pourcentages d'accessibilité PMR pour la France métropolitaine
pourcentage_accessibilite_metropole = (accessibilite_metropole / accessibilite_metropole.sum()) * 100
```

[55]:
```
fig, axs = plt.subplots(1, 2, figsize=(15, 10))

# Graphique pour les Hauts-de-France
axs[0].bar(pourcentage_accessibilite_hdf.index, pourcentage_accessibilite_hdf, color='lightgreen')
axs[0].set_title('Accessibilité PMR dans les Hauts-de-France')
axs[0].set_xlabel('Catégorie d\'accessibilité')
axs[0].set_ylabel('Pourcentage (%)')
axs[0].tick_params(axis='x', rotation=90)

# Graphique pour la France métropolitaine
axs[1].bar(pourcentage_accessibilite_metropole.index, pourcentage_accessibilite_metropole, color='lightblue')
axs[1].set_title('Accessibilité PMR en France métropolitaine')
axs[1].set_xlabel('Catégorie d\'accessibilité')
axs[1].set_ylabel('Pourcentage (%)')
axs[1].tick_params(axis='x', rotation=90)

plt.tight_layout()
plt.show()
```



Dans les Hauts-de-France, une proportion significative des logements a des données d'accessibilité non disponibles (plus de 40 %), tandis qu'une part importante est classée comme non accessible et non adaptée aux fauteuils roulants (plus de 40 %).

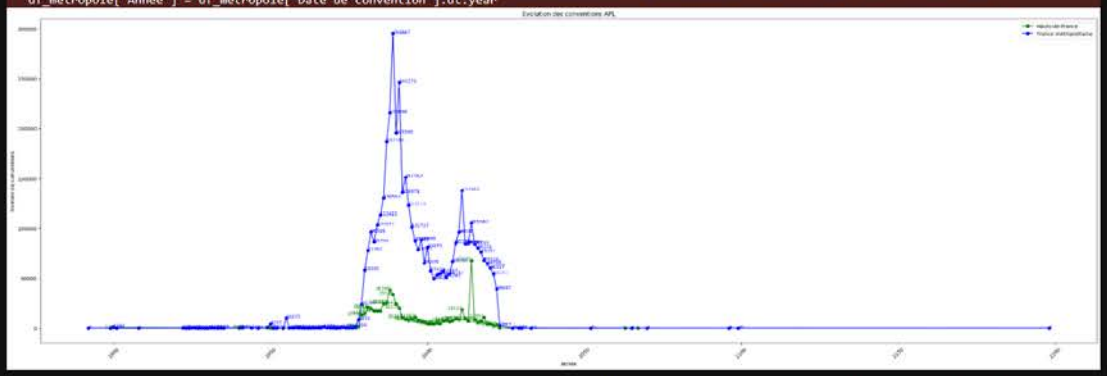
La France métropolitaine présente un profil quasi similaire, mais avec une proportion encore plus élevée de logements non accessibles et non adaptés aux fauteuils roulants. Cela suggère que, bien que l'accessibilité PMR soit un défi à l'échelle régionale et nationale.

Évolution des conventions APL

```
df_hdf['Date de convention'] = pd.to_datetime(df_hdf['Date de convention'], errors='coerce')
```

```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3893676868.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Date de convention'] = pd.to_datetime(df_hdf['Date de convention'], errors='coerce')
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3893676868.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_metropole['Date de convention'] = pd.to_datetime(df_metropole['Date de convention'], errors='coerce')
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3893676868.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Année'] = df_hdf['Date de convention'].dt.year
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3893676868.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_metropole['Année'] = df_metropole['Date de convention'].dt.year
```



Ici, on a le droit à une tendance similaire entre les deux zones géographiques, avec une augmentation des conventions APL jusqu'à franchir un pic, suivie d'une baisse progressive. Les Hauts-de-France semblent avoir suivi la tendance nationale, avec un nombre légèrement inférieur de conventions APL. Cela pourrait indiquer une dépendance accrue à l'APL au niveau national, tout en reflétant une stabilisation ou une diminution après une période de forte croissance.

Impact des politiques urbaines

[60]:
```
# Calcul de la proportion de logements en QPV
proportion_qpv = df_log['QPV_LIBELLE'].value_counts(normalize=True) * 100
proportion_qpv
```

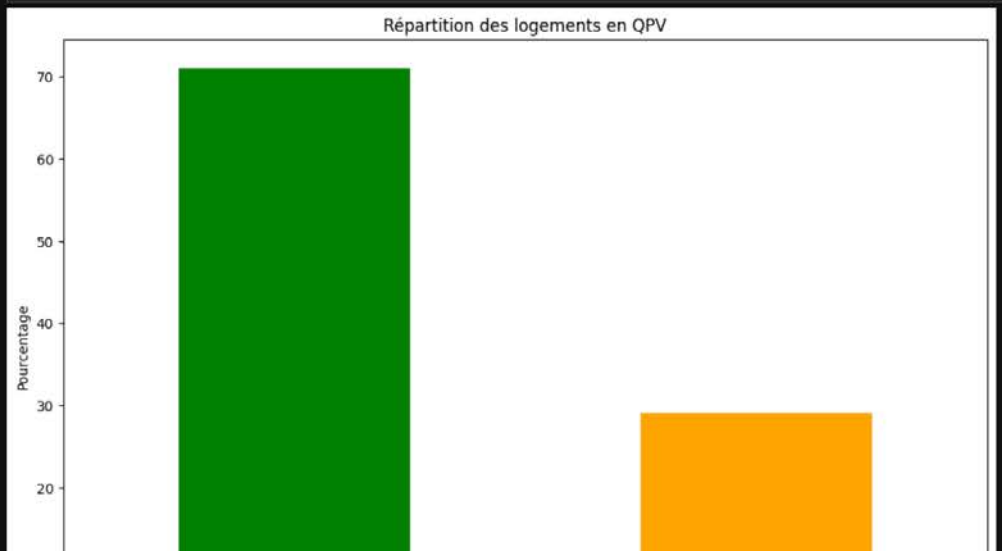
[60]:
```
QPV_LIBELLE
Non    70.92604
Oui    29.07396
Name: proportion, dtype: float64
```

[63]:
```
# Calcul de la proportion de logements en ZUS
proportion_zus = df_log['Code ZUS'].value_counts(normalize=True) * 100

# Calcul de la proportion de logements en ZFU
proportion_zfu = df_log['Code ZFU'].value_counts(normalize=True) * 100

# Calcul de la proportion de logements en QVA
proportion_qva = df_log['Code QVA'].value_counts(normalize=True) * 100
```

[64]:
```
# Comparaison de la proportion de logements individuels et collectifs en QPV et hors QPV
logements_qpv = df_log[df_log['QPV_LIBELLE'] == 'Oui']
logements_hors_qpv = df_log[df_log['QPV_LIBELLE'] == 'Non']

# Calcul des proportions de types de logements
proportion_types_qpv = logements_qpv['TYPECONST_LIBELLE'].value_counts(normalize=True) * 100
proportion_types_hors_qpv = logements_hors_qpv['TYPECONST_LIBELLE'].value_counts(normalize=True) * 100
```

[65]:
```
# Comparaison de la classe DPE des logements en QPV et hors QPV
proportion_dpe_qpv = logements_qpv['Etiquette DPE Energie'].value_counts(normalize=True) * 100
proportion_dpe_hors_qpv = logements_hors_qpv['Etiquette DPE Energie'].value_counts(normalize=True) * 100
```

[68]:
```
plt.figure(figsize=(12, 8))
proportion_qpv.plot(kind='bar', color=['green', 'orange'])
plt.title('Répartition des logements en QPV')
plt.xlabel('Présence en QPV')
plt.ylabel('Pourcentage')
plt.show()
```



Analyse des surfaces habitables

[70]:
```
df_hdf.head(1)
```

[70]:

	REG_CODE	REG_LIBELLE	DEP_CODE	DEP_LIBELLE	EPCI_CODE	EPCI_LIBELLE	DROIT_CODE	DROIT_LIBELLE	DEPCOM_CODE	DEPCOM_LIBELLE	Code Postal
48875	32	Hauts-de-France	02	Aisne	200071785	CA Chauny-Tergnier-La Fère	1	pleine propriété	02001	Abbécourt	02300

[73]:
```
# Conversion de la colonne Surface habitable en type numérique
df_hdf['Surface habitable'] = pd.to_numeric(df_hdf['Surface habitable'], errors='coerce')

# Vérification des valeurs manquantes
missing_values = df_hdf['Surface habitable'].isna().sum()
print(f'Nombre de valeurs manquantes dans Surface habitable: {missing_values}')

# Traitement des valeurs manquantes en les remplaçant par la médiane (ou une autre stratégie)
df_hdf['Surface habitable'] = df_hdf['Surface habitable'].fillna(df_hdf['Surface habitable'].median())

# Vérification des valeurs aberrantes (par exemple, surfaces très petites ou très grandes)
aberrant_values = df_hdf[(df_hdf['Surface habitable'] < 10) | (df_hdf['Surface habitable'] > 500)]
print(f'Valeurs aberrantes: {aberrant_values.shape[0]} logements')
```

```
Nombre de valeurs manquantes dans Surface habitable: 0
Valeurs aberrantes: 3 logements
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3824379562.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Surface habitable'] = pd.to_numeric(df_hdf['Surface habitable'], errors='coerce')
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\3824379562.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Surface habitable'] = df_hdf['Surface habitable'].fillna(df_hdf['Surface habitable'].median())
```

[82]:
```
# Définir les tranches de surface
bins = [0, 30, 50, 70, 90, 110, float('inf')]
labels = ['<30 m²', '30-50 m²', '50-70 m²', '70-90 m²', '90-110 m²', '>110 m²']

# Créer une colonne pour les tranches de surface
df_hdf['Tranche de surface'] = pd.cut(df_hdf['Surface habitable'], bins=bins, labels=labels, include_lowest=True)

# Calculer la répartition des logements par tranche de surface
distribution_surface = df_hdf['Tranche de surface'].value_counts(normalize=True) * 100
distribution_surface.round(2)
```

```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\2021229372.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Tranche de surface'] = pd.cut(df_hdf['Surface habitable'], bins=bins, labels=labels, include_lowest=True)
```

[82]:
```
Tranche de surface
50-70 m²     40.34
70-90 m²     33.31
30-50 m²     13.96
90-110 m²     8.76
<30 m²        2.25
>110 m²       1.39
Name: proportion, dtype: float64
```

[80]:
```
# Répartition des surfaces habitables par département
distribution_par_departement = df_hdf.groupby('DEP_LIBELLE')['Tranche de surface'].value_counts(normalize=True).unstack().fillna(0) * 100
distribution_par_departement.round(2)
```

[80]:

Tranche de surface	<30 m²	30-50 m²	50-70 m²	70-90 m²	90-110 m²	>110 m²
DEP_LIBELLE						
Aisne	2.05	15.62	42.19	29.48	9.55	1.11
Nord	2.79	15.60	39.09	32.26	8.67	1.60
Oise	2.24	15.62	41.92	31.39	7.98	0.84
Pas-de-Calais	0.92	9.07	41.28	37.80	9.38	1.55
Somme	3.96	17.48	40.58	29.90	7.53	0.55

[81]:
```
# Répartition des surfaces habitables par type de logement
distribution_par_type = df_hdf.groupby('TYPECONST_LIBELLE')['Tranche de surface'].value_counts(normalize=True).unstack().fillna(0) * 100
distribution_par_type.round(2)
```

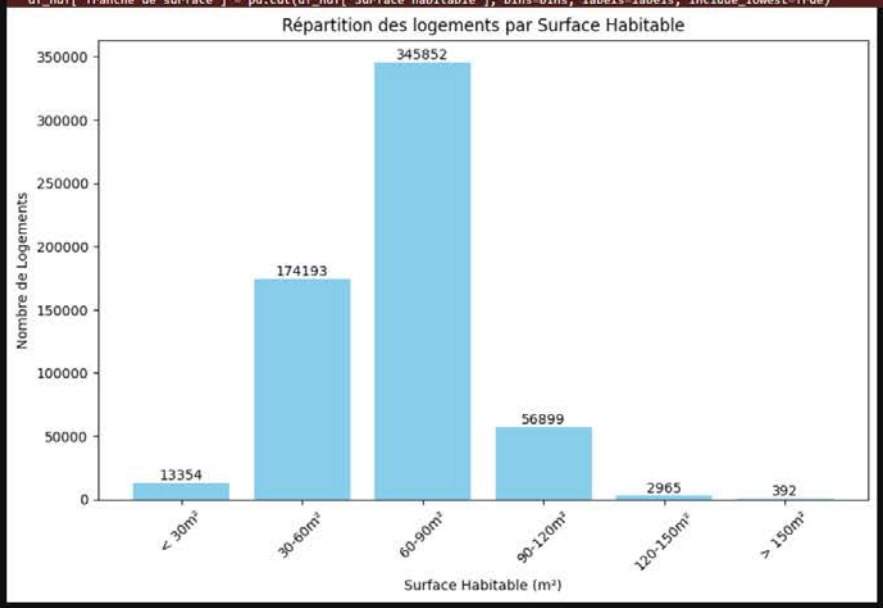
[81]:

Tranche de surface	<30 m²	30-50 m²	50-70 m²	70-90 m²	90-110 m²	>110 m²
TYPECONST_LIBELLE						
collectif	1.48	18.35	47.35	27.82	4.44	0.55

individuel	0.20	6.81	30.26	43.66	16.25	2.82
logement étudiant	86.77	10.85	1.31	0.82	0.22	0.03

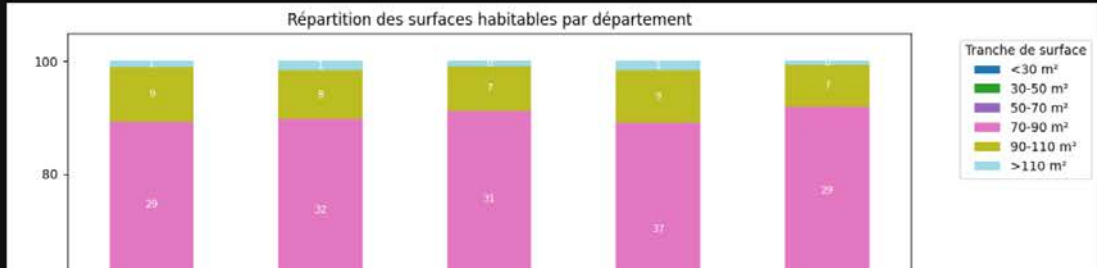
[94]:
```
# Création des intervalles de surface
bins = [0, 30, 60, 90, 120, 150, float('inf')]
labels = ['< 30m²', '30-60m²', '60-90m²', '90-120m²', '120-150m²', '> 150m²']

# Catégorisation des logements en fonction de la surface habitable
df_hdf['Tranche de surface'] = pd.cut(df_hdf['Surface habitable'], bins=bins, labels=labels, include_lowest=True)

# Calcul de la distribution
distribution_surface = df_hdf['Tranche de surface'].value_counts().sort_index()

# Création de l'histogramme
plt.figure(figsize=(10, 6))
bars = plt.bar(distribution_surface.index, distribution_surface.values, color='skyblue')

# Ajout des étiquettes pour chaque barre, centrées au-dessus
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), ha='center', va='bottom')  # ha='center' pour centrer horizontalement


plt.title('Répartition des logements par Surface Habitable')
plt.xlabel('Surface Habitable (m²)')
plt.ylabel('Nombre de Logements')
plt.xticks(rotation=45)
plt.show()
```

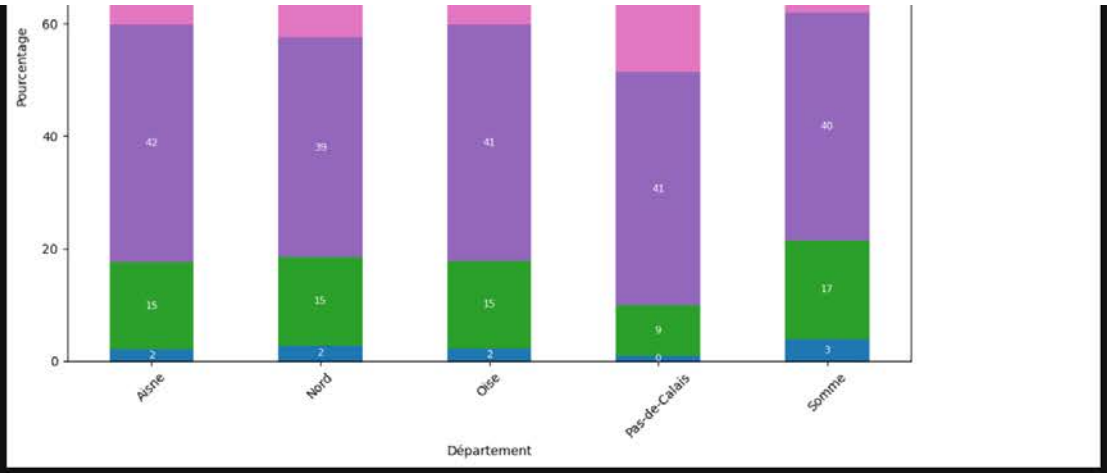
```
C:\Users\Laetitia_Deken\AppData\Local\Temp\ipykernel_33052\4003902549.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_hdf['Tranche de surface'] = pd.cut(df_hdf['Surface habitable'], bins=bins, labels=labels, include_lowest=True)
```



La majorité des logements ont une surface comprise entre 60 et 90 m² (345852), suivis par ceux de 30 à 60 m². Les logements de moins de 30 m², ainsi que ceux de plus de 120 m², sont nettement moins nombreux. Cela indique une préférence marquée pour des logements de taille moyenne dans la région, tandis que les très petites et très grandes surfaces sont moins courantes.

[92]:
```
ax = distribution_par_departement.plot(kind='bar', stacked=True, figsize=(12, 8), colormap='tab20')

# Ajout des étiquettes
for rect in ax.patches:
    height = rect.get_height()
    if height > 0:  # Pour éviter les étiquettes pour les barres avec une hauteur de 0
        ax.text(rect.get_x() + rect.get_width() / 2, rect.get_y() + height / 2,
                f'{int(height)}', ha='center', va='center', fontsize=8, color='white')

plt.title('Répartition des surfaces habitables par département')
plt.xlabel('Département')
plt.ylabel('Pourcentage')
plt.xticks(rotation=45)
plt.legend(title='Tranche de surface', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```





Typologie des bâtiments

[23]:
```
# Répartition des typologies de bâtiments pour les Hauts-de-France
hdf_typologie = df_log[df_log['REG_LIBELLE'] == 'Hauts-de-France']['TYPECONST_LIBELLE'].value_counts(normalize=True) * 100

# Répartition des typologies de bâtiments pour la France métropolitaine (hors DOM-TOM)
metropole_typologie = df_log[~df_log['REG_LIBELLE'].isin(['Guadeloupe', 'Martinique', 'Guyane', 'La Réunion', 'Mayotte'])]['TYPECONST_LIBELLE'].value_cou
```

[25]:
```
# Fusion des résultats pour comparaison
typologie_comparaison = pd.DataFrame({
    'Hauts-de-France': hdf_typologie,
    'France métropolitaine': metropole_typologie
})

# Création d'un graphique en barres
ax = typologie_comparaison.plot(kind='bar', figsize=(12, 8), color=['blue', 'orange'])
plt.title('Répartition des typologies de bâtiments')
plt.xlabel('Type de bâtiment')
plt.ylabel('Pourcentage')
plt.xticks(rotation=45)
plt.legend(title='Région')

# Ajout des étiquettes centrées pour chaque barre
for p in ax.patches:
    ax.annotate(f'{p.get_height():.1f}%',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10),
                textcoords='offset points', color=p.get_facecolor())

plt.tight_layout()
plt.show()
```