exploration

October 21, 2025

1 Predicting property sale prices in France: a study based on the DVF dataset

1.1 Exploratory notebook

1.2 Necessary libraires import

[2]:
```
# Importing core libraries for data manipulation and visualization
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import missingno as msno
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import geopandas as gpd
import imageio.v2 as iio
import matplotlib.dates as mdates

# Plotly (Advanced graphs library) configuration
import plotly.io as pio
pio.renderers.default = "notebook"

# Warning management (to keep output clean)
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

1.2.1 Load DVF Dataset

[3]:
```
# Load the DVF dataset as DataFrame
df_dvf = pd.read_csv('dvf.csv', low_memory=False)

# Display the first 10 rows
df_dvf.head(10)
```

[3]:
```
  id_mutation date_mutation  numero_disposition nature_mutation  \
0      2020-1    2020-01-07                   1           Vente
1      2020-2    2020-01-02                   1           Vente
2      2020-2    2020-01-02                   1           Vente
3      2020-2    2020-01-02                   1           Vente
4      2020-2    2020-01-02                   1           Vente
5      2020-2    2020-01-02                   1           Vente
6      2020-3    2020-01-07                   1           Vente
7      2020-4    2020-01-07                   1           Vente
8      2020-4    2020-01-07                   1           Vente
9      2020-5    2020-01-09                   1           Vente

   valeur_fonciere  adresse_numero adresse_suffixe        adresse_nom_voie  \
0           8000.0             NaN             NaN                FORTUNAT
1           2175.0             NaN             NaN  TERRES DES CINQ SAULES
2           2175.0             NaN             NaN      BOIS DU CHAMP RION
3           2175.0             NaN             NaN             EN COROBERT
4           2175.0             NaN             NaN  TERRES DES CINQ SAULES
5           2175.0             NaN             NaN  TERRES DES CINQ SAULES
6          75000.0             NaN             NaN    RUE DE LA CHARTREUSE
7            123.0             NaN             NaN           CHAMP PORTIER
8            123.0             NaN             NaN           CHAMP PORTIER
9          72000.0             NaN             NaN                 CHAMPEL

  adresse_code_voie  code_postal  … type_local surface_reelle_bati  \
0              B063       1250.0  …        NaN                 NaN
1              B124       1290.0  …        NaN                 NaN
2              B006       1290.0  …        NaN                 NaN
3              B025       1290.0  …        NaN                 NaN
4              B124       1290.0  …        NaN                 NaN
5              B124       1290.0  …        NaN                 NaN
6              0064       1960.0  …        NaN                 NaN
7              B041       1370.0  …        NaN                 NaN
8              B041       1370.0  …        NaN                 NaN
9              B034       1270.0  …        NaN                 NaN

  nombre_pieces_principales  code_nature_culture    nature_culture  \
0                       NaN                    T            terres
1                       NaN                   BT   taillis simples
2                       NaN                    T            terres
3                       NaN                    T            terres
4                       NaN                    T            terres
5                       NaN                    T            terres
6                       NaN                   AB  terrains a bâtir
7                       NaN                    S              sols
8                       NaN                    S              sols
9                       NaN                    J           jardins
```

```
  code_nature_culture_speciale nature_culture_speciale surface_terrain  \
0                          NaN                     NaN          1061.0
1                          NaN                     NaN            85.0
2                          NaN                     NaN          1115.0
3                          NaN                     NaN          1940.0
4                          NaN                     NaN          1148.0
5                          NaN                     NaN          2960.0
6                          NaN                     NaN           610.0
7                          NaN                     NaN            55.0
8                          NaN                     NaN            68.0
9                          NaN                     NaN           328.0

  longitude   latitude
0  5.323532  46.171941
1  4.893454  46.251858
2  4.900210  46.235277
3  4.882112  46.246554
4  4.894481  46.251841
5  4.894616  46.251941
6  5.226216  46.184570
7  5.344427  46.263955
8  5.343896  46.263803
9  5.350547  46.380898

[10 rows x 40 columns]
```

1.2.2 Inspect Dataset properties

Thanks to the **info** and **describe** methods, it is possible to quickly view the data contained in our dataset (entire CSV equivalent to 3GB). - The first method allows us to access the column names and their associated value types. - The second method allows us to view the orders of magnitude of the different columns.

[3]:
```python
# Check data types, non-null counts, and overall structure
df_dvf.info()

# Summary statistics for numerical columns (mean, std, min, max, and quartiles
↪to understand data distribution)
df_dvf.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20133668 entries, 0 to 20133667
Data columns (total 40 columns):
 #   Column                        Dtype
---  ------                        -----
 0   id_mutation                   object
 1   date_mutation                 object
```

```
 2   numero_disposition           int64
 3   nature_mutation              object
 4   valeur_fonciere              float64
 5   adresse_numero               float64
 6   adresse_suffixe              object
 7   adresse_nom_voie             object
 8   adresse_code_voie            object
 9   code_postal                  float64
 10  code_commune                 object
 11  nom_commune                  object
 12  code_departement             object
 13  ancien_code_commune          float64
 14  ancien_nom_commune           object
 15  id_parcelle                  object
 16  ancien_id_parcelle           object
 17  numero_volume                object
 18  lot1_numero                  object
 19  lot1_surface_carrez          float64
 20  lot2_numero                  object
 21  lot2_surface_carrez          float64
 22  lot3_numero                  object
 23  lot3_surface_carrez          float64
 24  lot4_numero                  object
 25  lot4_surface_carrez          float64
 26  lot5_numero                  object
 27  lot5_surface_carrez          float64
 28  nombre_lots                  int64
 29  code_type_local              float64
 30  type_local                   object
 31  surface_reelle_bati          float64
 32  nombre_pieces_principales    float64
 33  code_nature_culture          object
 34  nature_culture               object
 35  code_nature_culture_speciale object
 36  nature_culture_speciale      object
 37  surface_terrain              float64
 38  longitude                    float64
 39  latitude                     float64
dtypes: float64(15), int64(2), object(23)
memory usage: 6.0+ GB
```

[3]:

```
       numero_disposition  valeur_fonciere  adresse_numero   code_postal  \
count        2.013367e+07     1.993772e+07    1.260362e+07  1.998002e+07
mean         1.234985e+00     1.529514e+06    7.150326e+02  5.000524e+04
std          7.899619e+00     1.684699e+07    2.017854e+03  2.739363e+04
min          1.000000e+00     1.000000e-02    1.000000e+00  1.000000e+03
25%          1.000000e+00     6.900000e+04    8.000000e+00  2.823000e+04
```

```
50%          1.000000e+00     1.650000e+05    2.500000e+01  4.914000e+04
75%          1.000000e+00     3.030000e+05    1.000000e+02  7.500700e+04
max          1.246000e+03     1.415000e+10    9.999000e+03  9.749000e+04

       ancien_code_commune  lot1_surface_carrez  lot2_surface_carrez  \
count           408.000000         1.761506e+06        557955.000000
mean          20094.698529         6.735296e+01            64.035822
std           18521.130057         2.082315e+02            71.815365
min           14666.000000         1.000000e-02             0.010000
25%           14666.000000         3.572000e+01            43.820000
50%           14666.000000         5.510000e+01            61.570000
75%           15031.000000         7.410000e+01            76.620000
max           85212.000000         9.614000e+03          8705.000000

       lot3_surface_carrez  lot4_surface_carrez  lot5_surface_carrez  \
count         61941.000000         14984.000000          5271.000000
mean             71.970848            84.467058            96.111413
std              98.255307           142.936682           196.619148
min               0.200000             0.340000             0.400000
25%              41.680000            39.617500            35.635000
50%              62.080000            67.360000            70.080000
75%              85.050000           100.370000           114.100000
max            6947.850000          6947.850000          6947.850000

        nombre_lots  code_type_local  surface_reelle_bati  \
count  2.013367e+07     1.183813e+07         6.983283e+06
mean   4.320833e-01     2.226831e+00         1.153680e+02
std    8.386084e-01     9.308073e-01         8.281463e+02
min    0.000000e+00     1.000000e+00         1.000000e+00
25%    0.000000e+00     1.000000e+00         5.000000e+01
50%    0.000000e+00     2.000000e+00         7.500000e+01
75%    1.000000e+00     3.000000e+00         1.050000e+02
max    2.360000e+02     4.000000e+00         5.934000e+05

       nombre_pieces_principales  surface_terrain     longitude      latitude
count               1.182539e+07     1.372763e+07  1.972417e+07  1.972417e+07
mean                1.863278e+00     2.863297e+03  2.354309e+00  4.610026e+01
std                 2.091897e+00     1.409837e+04  6.379150e+00  5.921900e+00
min                 0.000000e+00     1.000000e+00 -6.315108e+01 -2.138654e+01
25%                 0.000000e+00     2.460000e+02  3.268020e-01  4.466372e+01
50%                 1.000000e+00     6.220000e+02  2.365491e+00  4.672001e+01
75%                 4.000000e+00     1.782000e+03  4.643447e+00  4.866416e+01
max                 1.980000e+02     1.072309e+07  5.583079e+01  5.108645e+01
```

[4]:

```python
print(df_dvf.columns) # Print dataframe columns names
```

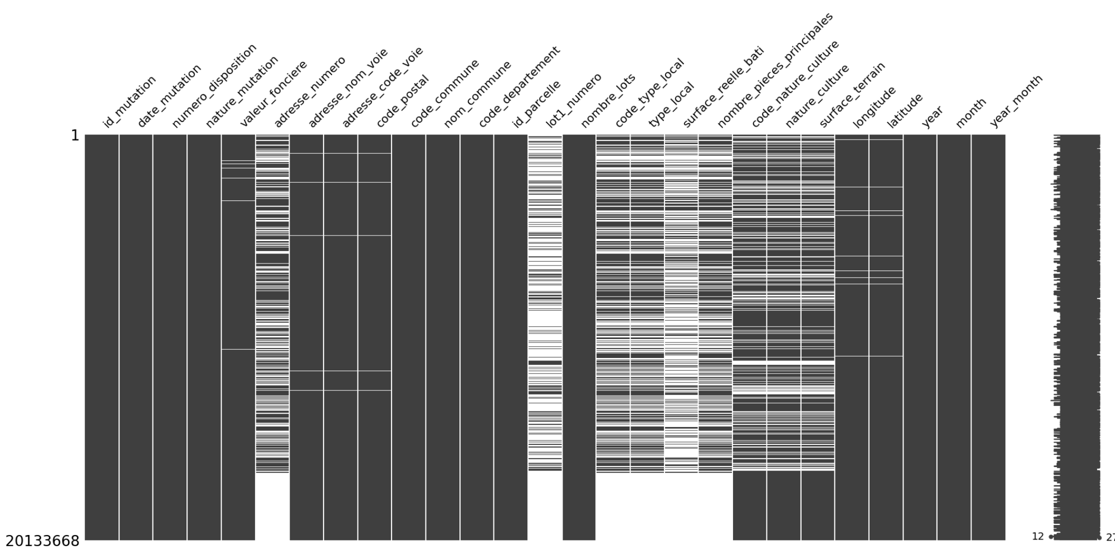
```
Index(['id_mutation', 'date_mutation', 'numero_disposition', 'nature_mutation',
       'valeur_fonciere', 'adresse_numero', 'adresse_suffixe',
```

```
 'adresse_nom_voie', 'adresse_code_voie', 'code_postal', 'code_commune',
 'nom_commune', 'code_departement', 'ancien_code_commune',
 'ancien_nom_commune', 'id_parcelle', 'ancien_id_parcelle',
 'numero_volume', 'lot1_numero', 'lot1_surface_carrez', 'lot2_numero',
 'lot2_surface_carrez', 'lot3_numero', 'lot3_surface_carrez',
 'lot4_numero', 'lot4_surface_carrez', 'lot5_numero',
 'lot5_surface_carrez', 'nombre_lots', 'code_type_local', 'type_local',
 'surface_reelle_bati', 'nombre_pieces_principales',
 'code_nature_culture', 'nature_culture', 'code_nature_culture_speciale',
 'nature_culture_speciale', 'surface_terrain', 'longitude', 'latitude'],
dtype='object')
```

1.2.3 Missing values overview

Thanks to the **missingno** library, it is possible to quickly view the null values contained in the columns as well as their total number. This allows us to easily identify whether it will be possible to delete columns or whether it will be necessary to evaluate this missing data.

[76]:
```
msno.matrix(df_dvf)
plt.show()
```



[6]:
```
df_dvf.isnull().sum() / len(df_dvf) * 100 # Calculate the percentage of missing␣
↪data for each column
```

[6]:
```
id_mutation                    0.000000
date_mutation                  0.000000
numero_disposition             0.000000
nature_mutation                0.000000
valeur_fonciere                0.973255
```

```
adresse_numero                  37.400254
adresse_suffixe                 95.517046
adresse_nom_voie                 0.761684
adresse_code_voie                0.757830
code_postal                      0.763164
code_commune                     0.000000
nom_commune                      0.000000
code_departement                 0.000000
ancien_code_commune             99.997974
ancien_nom_commune              99.997974
id_parcelle                      0.000000
ancien_id_parcelle              99.999930
numero_volume                   99.770166
lot1_numero                     68.647536
lot1_surface_carrez             91.250943
lot2_numero                     91.009393
lot2_surface_carrez             97.228746
lot3_numero                     98.398518
lot3_surface_carrez             99.692351
lot4_numero                     99.486571
lot4_surface_carrez             99.925577
lot5_numero                     99.779335
lot5_surface_carrez             99.973820
nombre_lots                      0.000000
code_type_local                 41.202304
type_local                      41.202304
surface_reelle_bati             65.315396
nombre_pieces_principales       41.265610
code_nature_culture             31.814173
nature_culture                  31.814173
code_nature_culture_speciale    95.707702
nature_culture_speciale         95.707702
surface_terrain                 31.817556
longitude                        2.033917
latitude                         2.033917
dtype: float64
```

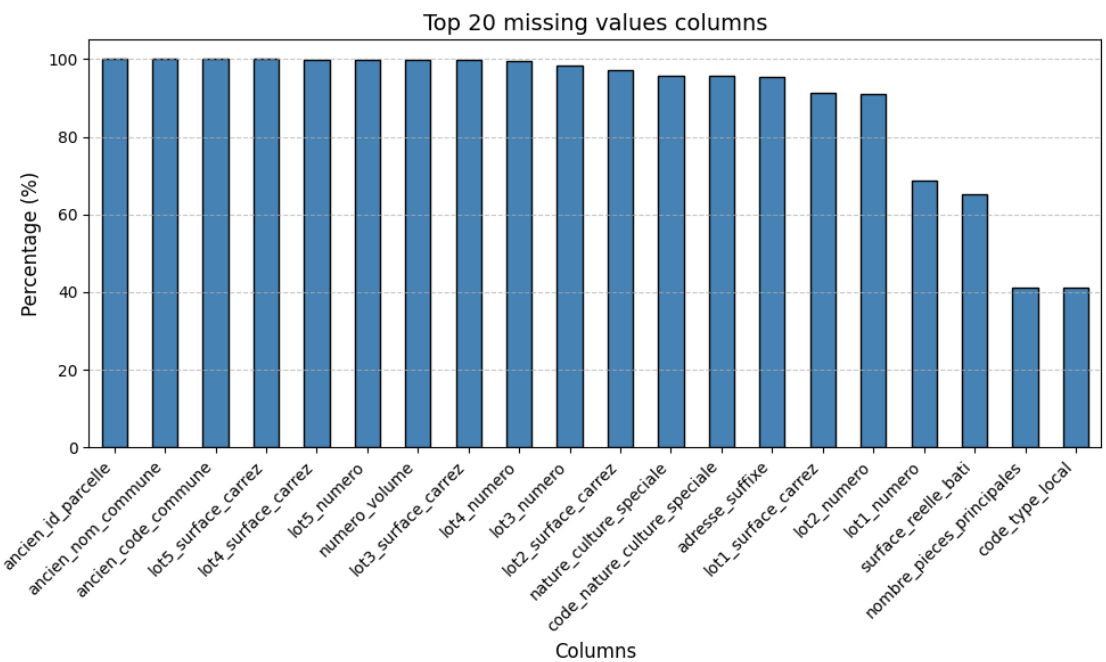
[7]:
```
missing_percent = (df_dvf.isnull().sum() / len(df_dvf) * 100).
  ↪sort_values(ascending=False)

# Take only the 15 columns with more missing values
top_missing = missing_percent.head(20)

# Barplot
plt.figure(figsize=(10,6))
top_missing.plot(kind='bar', color='steelblue', edgecolor='black')
```

```
plt.title('Top 20 missing values columns', fontsize=14)
plt.ylabel('Percentage (%)', fontsize=12)
plt.xlabel('Columns', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)

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



To directly access the names of the 'deletable' columns, i.e. those with more than 90% missing values, we save them in a list.

[8]:
```
# Delete columns
df_dvf = df_dvf.loc[:, df_dvf.isnull().mean() < 0.9]

df_dvf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20133668 entries, 0 to 20133667
Data columns (total 24 columns):
 #   Column                      Dtype
---  ------                      -----
 0   id_mutation                 object
 1   date_mutation               object
 2   numero_disposition          int64
 3   nature_mutation             object
```

```
 4   valeur_fonciere            float64
 5   adresse_numero             float64
 6   adresse_nom_voie           object
 7   adresse_code_voie          object
 8   code_postal                float64
 9   code_commune               object
 10  nom_commune                object
 11  code_departement           object
 12  id_parcelle                object
 13  lot1_numero                object
 14  nombre_lots                int64
 15  code_type_local            float64
 16  type_local                 object
 17  surface_reelle_bati        float64
 18  nombre_pieces_principales  float64
 19  code_nature_culture        object
 20  nature_culture             object
 21  surface_terrain            float64
 22  longitude                  float64
 23  latitude                   float64
dtypes: float64(9), int64(2), object(13)
memory usage: 3.6+ GB
```

2 Correlation matrix

The correlation matrix provides an overview of the linear relationships between the numerical variables in the DVF dataset. Correlation values range from -1 (negative relationship) to +1 (positive relationship).

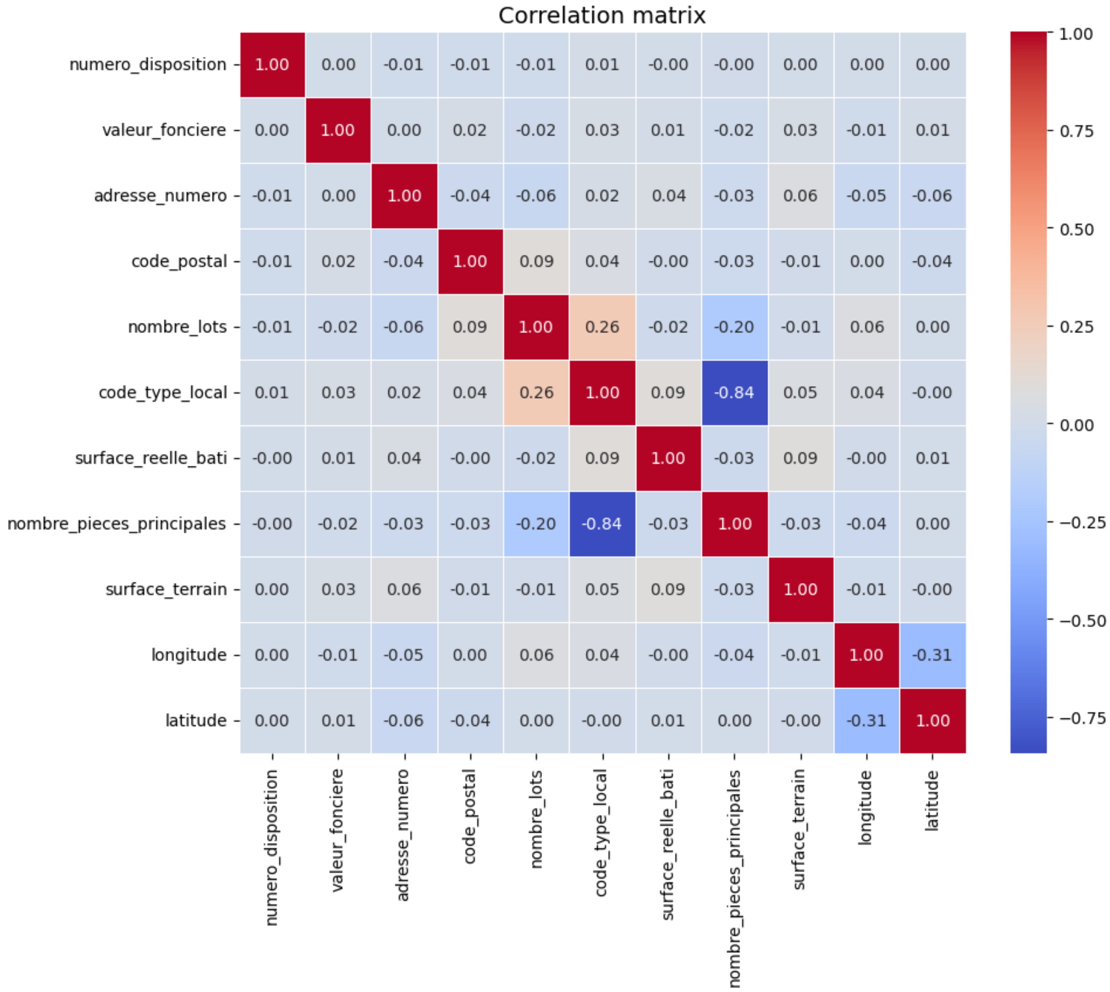
Most variables show weak linear relationships, as most correlations are near zero. **valeur_fonciere** (property value) has no strong correlation with other features, suggesting that prices depend on multiple non-linear factors such as location and property type. The strongest correlation is negative (-0.84) between **code_type_local** and **nombre_pieces_principales**, meaning that property type strongly influences room count. Overall, the dataset shows low multicollinearity, good for modeling, but linear models may not fully capture price dynamics.

[9]:
```python
corr = df_dvf.corr(numeric_only=True)

# Visualize the correlation matrix
plt.figure(figsize=(10,8))
sns.heatmap(corr,
            cmap='coolwarm',
            annot=True,
            fmt=".2f",
            linewidths=0.5,
            cbar=True)

plt.title("Correlation matrix", fontsize=14)
```

```
plt.show()
```

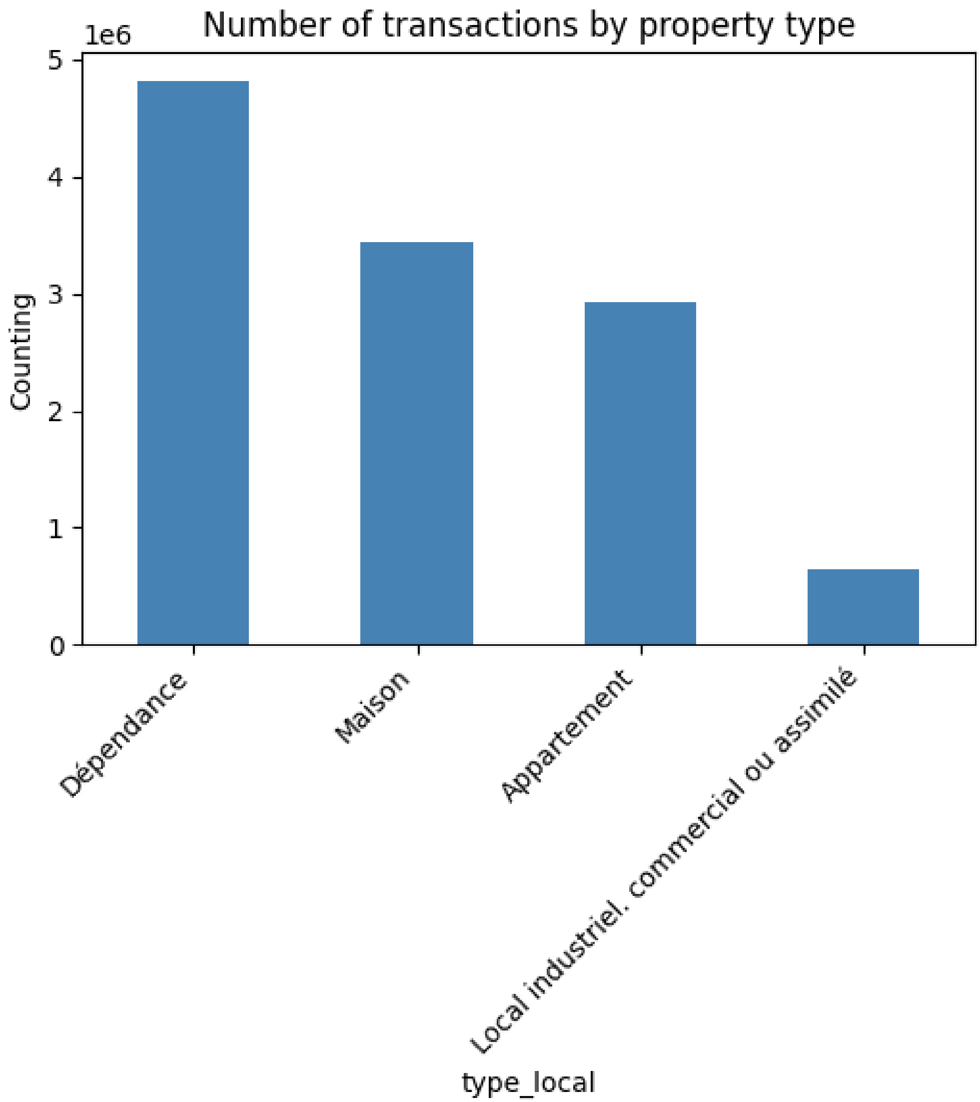


2.1 Property Type Distribution

The dataset shows that most transactions involve outbuildings and houses, while apartments and commercial premises are less frequent. This reflects the predominance of residential property sales in the DVF dataset.

[22]:
```
# Visualize how many transactions correspond to each property category/type
df_dvf['type_local'].value_counts().plot(kind='bar', figsize=(6,4),␣
    ↪color='steelblue')

plt.title("Number of transactions by property type")
plt.ylabel("Count")
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.show()
```



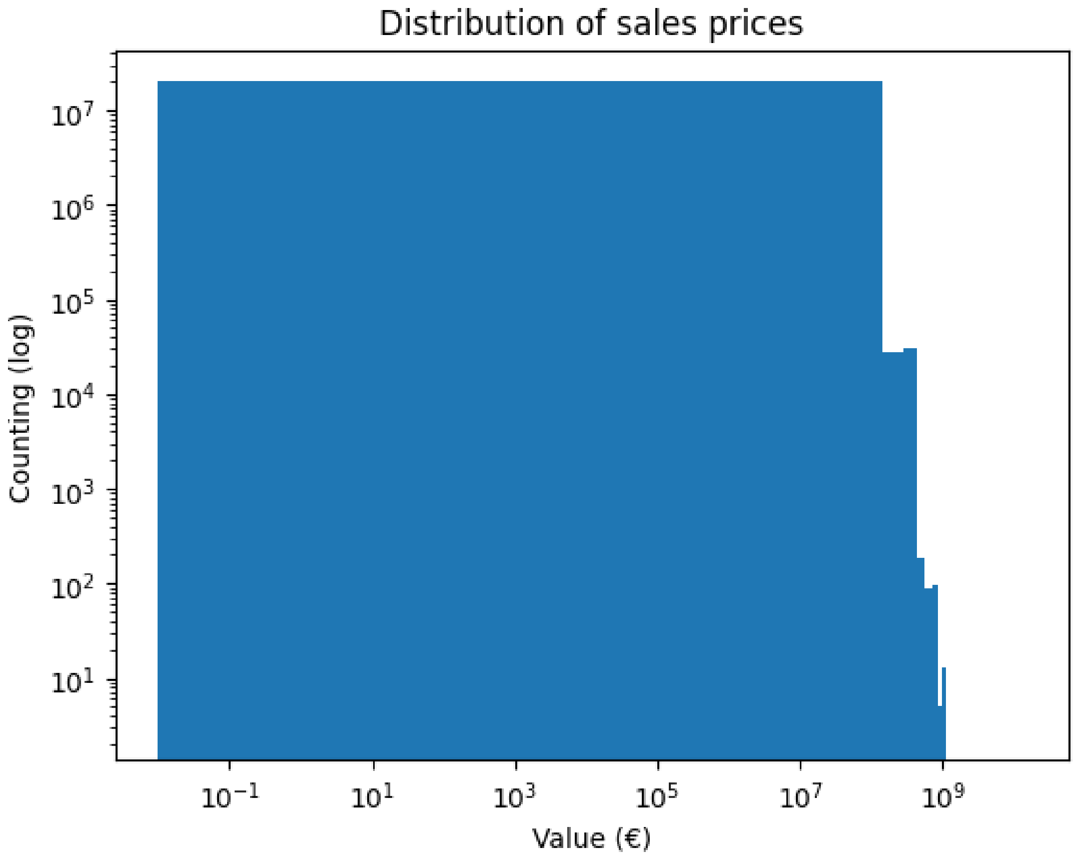
2.1.1 Distribution of Sales Prices

The distribution of property sale prices is highly skewed, with most transactions involving relatively low to medium values. A few extreme outliers correspond to high-value properties. The **logarithmic scale** highlights the strong asymmetry typical of real estate markets.

[19]:
```
# Plot histogram
plt.hist(df_dvf['valeur_fonciere'], bins=100, log=True)

plt.xscale('log')
plt.xlabel("Value (€)")
plt.ylabel("Count (log scale)")
plt.title("Distribution of sales prices")
```

```
plt.show()
```

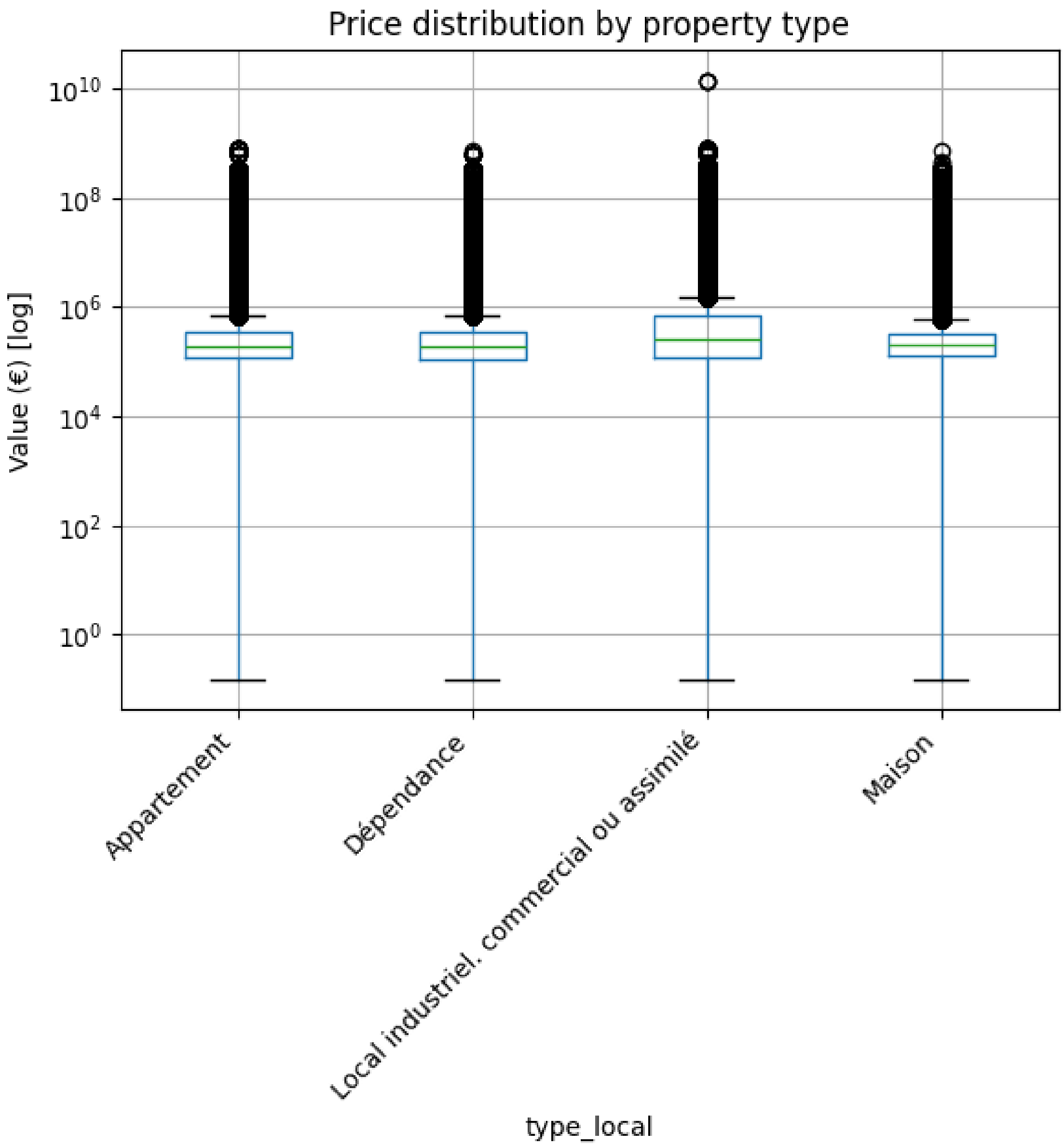


2.1.2 Price Distribution by Property Type

The boxplot shows that property prices vary widely across all categories, with numerous high-value outliers. Houses and apartments have comparable median prices, while commercial and industrial properties show higher variability. The strong skew confirms the presence of extreme values.

[20]:
```
# Use boxplots to compare the sale price ranges across property categories
df_dvf.boxplot(column='valeur_fonciere', by='type_local')

plt.yscale('log')  # Log scale to handle extreme price variations
plt.ylabel("Value (€) [log]")
plt.title("Price distribution by property type")
plt.suptitle("")  # Remove default title
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.show()
```



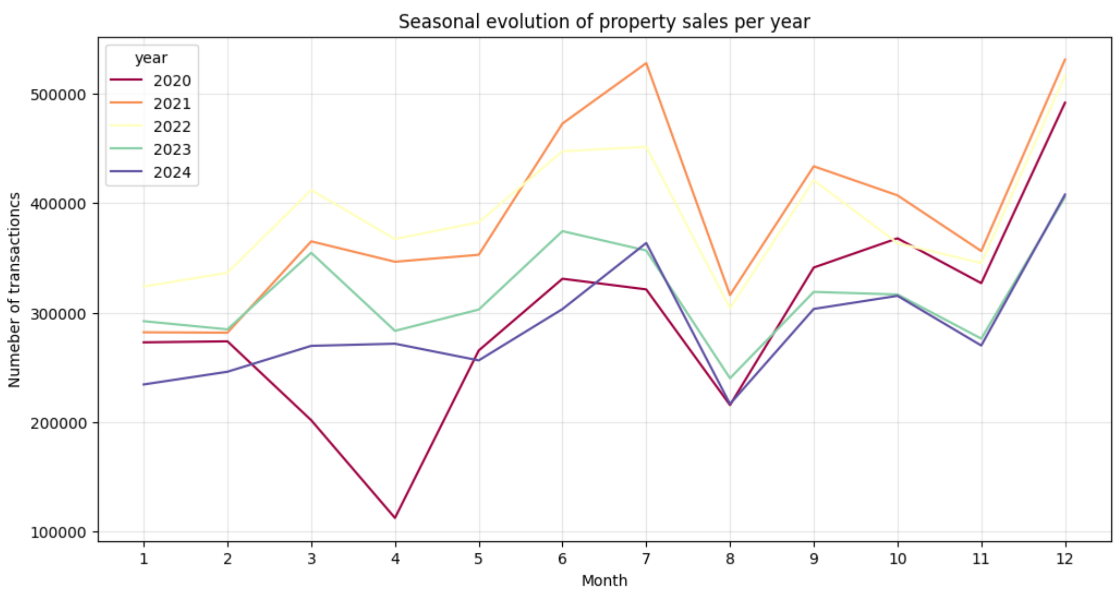
2.1.3 Seasonal plot

The number of transactions follows a clear seasonal pattern, typically peaking in summer and at the end of the year. Activity dropped significantly in 2020 probably due to the COVID-19 pandemic but gradually recovered in the following years

[17]:
```
# Extract year and month
df_dvf['date_mutation'] = pd.to_datetime(df_dvf['date_mutation'],␣
 ↪errors='coerce')
df_dvf['year'] = df_dvf['date_mutation'].dt.year
df_dvf['month'] = df_dvf['date_mutation'].dt.month
```

```
# Count the number of transactions by (year, month)
monthly_sales = df_dvf.groupby(['year', 'month']).size().
↪reset_index(name='count')

# Line plots: seasonality separated by year
plt.figure(figsize=(12,6))
sns.lineplot(
    data=monthly_sales,
    x='month', y='count', hue='year', palette='Spectral'
)
plt.title("Seasonal evolution of property sales per year")
plt.xlabel("Month")
plt.ylabel("Numeber of transactioncs")
plt.xticks(range(1,13))
plt.grid(alpha=0.3)
plt.show()
```

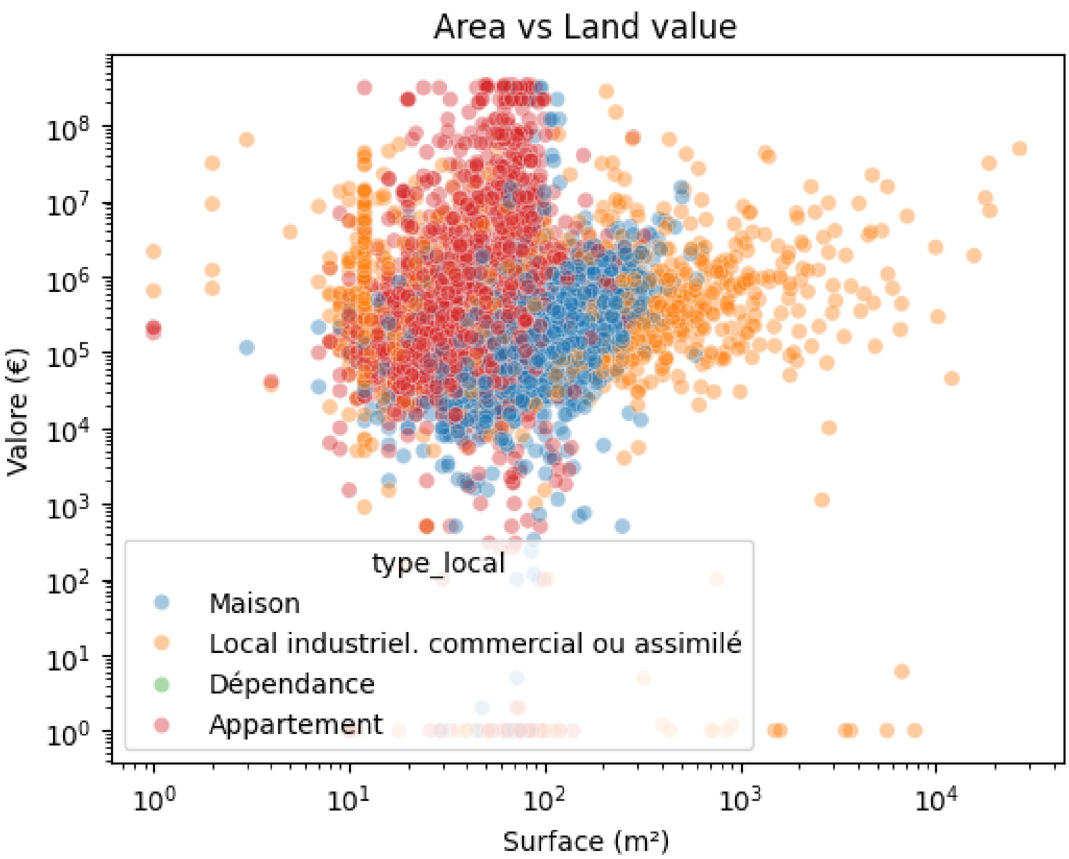


2.1.4 Impact of Surface Area on Property Prices

The scatter plot shows a positive relationship between surface area and property value, though with high dispersion. Larger properties generally sell for higher prices, but type differences and location effects create substantial variability.

[28]:
```
# Scatter plot to visualize how surface area relates to property price
sns.scatterplot(
    data=df_dvf.sample(50000),
```

```
    x='surface_reelle_bati', y='valeur_fonciere', hue='type_local', alpha=0.4
)
plt.xscale('log'); plt.yscale('log')
plt.title("Area vs Land value") #log scale
plt.xlabel("Surface (m²)"); plt.ylabel("Valore (€)")
plt.show()
```



2.1.5 Average Monthly Sale Price Over Time

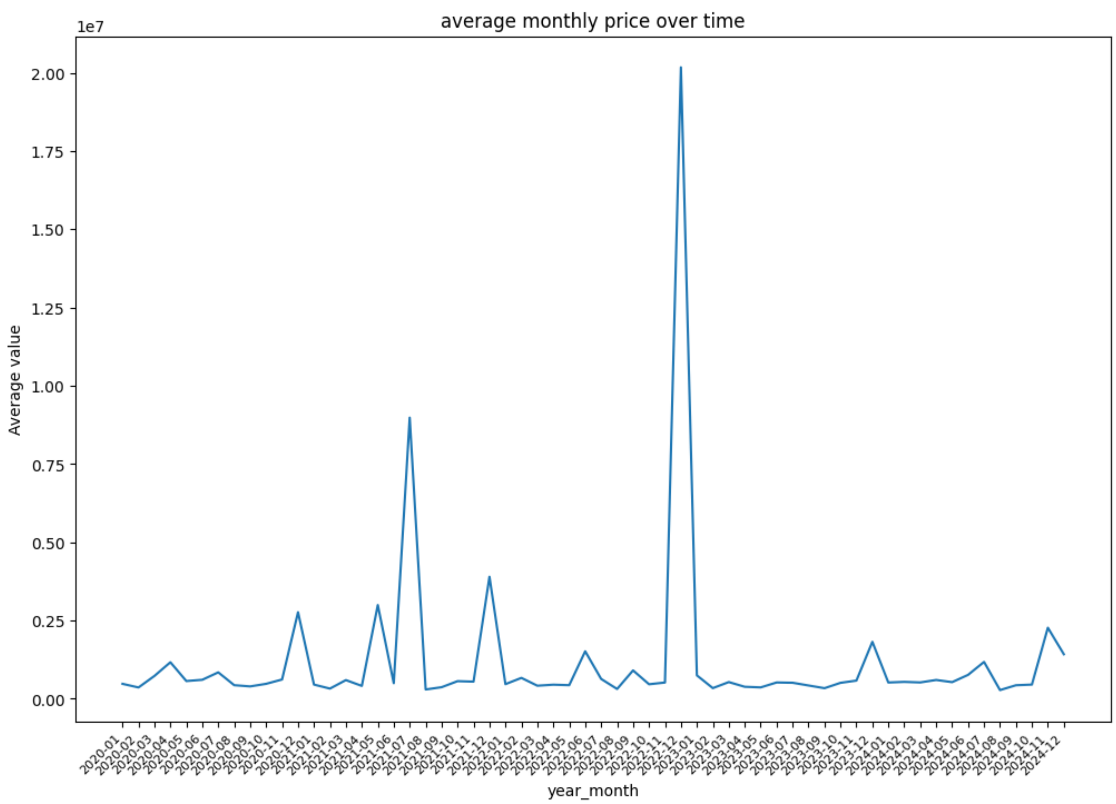
The average property value shows irregular fluctuations over time, with several extreme spikes likely caused by outlier transactions. These anomalies suggest the need for data cleaning or outlier filtering before modeling price trends.

[43]:
```
# To track market changes over time, convert date to time and total mean sale␣
↪prices by month.
df_dvf['date_mutation'] = pd.to_datetime(df_dvf['date_mutation'],␣
↪errors='coerce')
df_dvf['year_month'] = df_dvf['date_mutation'].dt.to_period('M')
```

```
monthly_price = (
    df_dvf.groupby('year_month')['valeur_fonciere']
     .mean()
     .reset_index()
)
monthly_price['year_month'] = monthly_price['year_month'].astype(str)

#  Line plot showing monthly average price evolution
plt.figure(figsize=(12,8))
sns.lineplot(data=monthly_price, x='year_month', y='valeur_fonciere')

plt.title("Average monthly price over time")
plt.ylabel("Average value (€)")
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.show()
```



2.1.6 Geographical Distribution of Property Sales in France

The geographical visualization highlights clear spatial patterns in real estate transactions across France. Most transactions are concentrated in Île-de-France, Auvergne-Rhône-Alpes, and Provence-Alpes-Côte d'Azur, which correspond to the country's most urbanized regions. The Paris metropolitan area in particular shows both a high density of transactions and the presence of some of the highest property values.

Coastal regions, especially along the French Riviera and the Atlantic coast, also display clusters of high-value properties, reflecting strong demand in touristic and luxury real estate markets. In contrast, central and rural regions exhibit lower transaction density and generally lower prices, consistent with smaller populations and less market activity.

Overall, the map confirms a strong geographical inequality in the French housing market, with major cities and coastal zones concentrating the most expensive transactions.

[6]:
```
# Create a geographic subset and log-transform property values
df_geo = df_dvf.dropna(subset=['longitude', 'latitude', 'valeur_fonciere']).
 ↪copy()
df_geo['valeur_fonciere_log'] = np.log10(df_geo['valeur_fonciere'])

# Get 10'000 samples from the dataset (to have a rapid overview)
df_geo = df_geo.sample(10000, random_state=42)

fig, axes = plt.subplots(1, 2, figsize=(18, 9),
                         subplot_kw={'projection': ccrs.PlateCarree()})
ax1, ax2 = axes

# LEFT subplot: scatter plot with map background
ax1.add_feature(cfeature.BORDERS, linestyle='-', linewidth=0.5)
ax1.add_feature(cfeature.COASTLINE, linewidth=0.7)
ax1.add_feature(cfeature.LAND, edgecolor='black', facecolor='lightgray')
ax1.add_feature(cfeature.OCEAN, facecolor='lightblue')
ax1.set_extent([-5, 9, 41, 51], crs=ccrs.PlateCarree())

# Scatter plot
sc = ax1.scatter(
    df_geo['longitude'],
    df_geo['latitude'],
    c=df_geo['valeur_fonciere_log'],
    cmap='inferno',
    s=15,
    alpha=0.6,
    edgecolor='none',
    transform=ccrs.PlateCarree()
)

# Add colorbar to the first subplot (to differentiate values)
cbar = fig.colorbar(sc, ax=ax1, fraction=0.046, pad=0.04)
cbar.set_label('property value', fontsize=11)

# Add plot title & labels
ax1.set_title('Geographical distribution of real estate sales in France (Sample␣
 ↪10K)', fontsize=13)
```

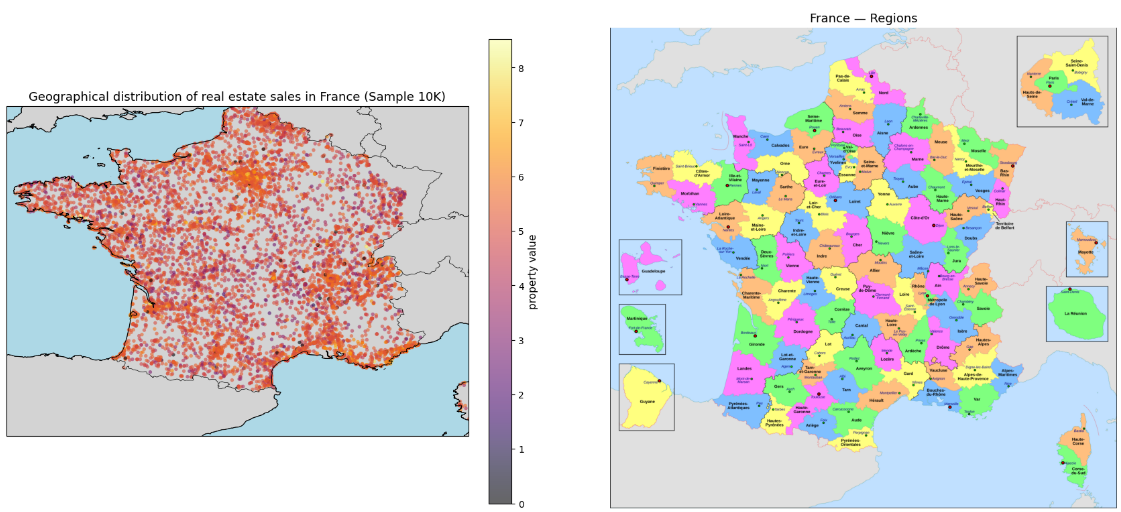
```
ax1.set_xlabel('Longitude')
ax1.set_ylabel('Latitude')

# RIGHT subplot: map image
fig.delaxes(ax2)  # remove the second cartopy subplot

ax_img = fig.add_axes([0.55, 0.1, 0.42, 0.8])

# Read and display the image
img = iio.imread("../docs/img/France_départementale.png")
ax_img.imshow(img)
ax_img.axis('off')
ax_img.set_title("France - Regions", fontsize=13)

# Adjust layout
plt.subplots_adjust(left=0.05, right=0.95, wspace=0.15)
plt.show()
```



2.1.7 Top 20 Departments by Number of Property Sales

The departments with the highest number of real estate transactions include Nord (59), Gironde (33), and Bouches-du-Rhône (13) — all of which are densely populated and economically active areas. These results align with national demographic patterns, as larger metropolitan regions tend to generate more sales activity. Departments such as Rhône (69), Alpes-Maritimes (06), and Haute-Garonne (31) also appear among the top 10, confirming strong real estate demand around major urban centers like Lyon, Nice, and Toulouse.

Overall, the chart highlights how property market activity is heavily concentrated in France's most urbanized regions.

[38]:
```
# Extract deparment list with relative code
dept_list = df_dvf[['code_departement', 'nom_commune']].drop_duplicates().
↪sort_values('code_departement')

# Print list
print(len(dept_list))
dept_list.head(10)
```

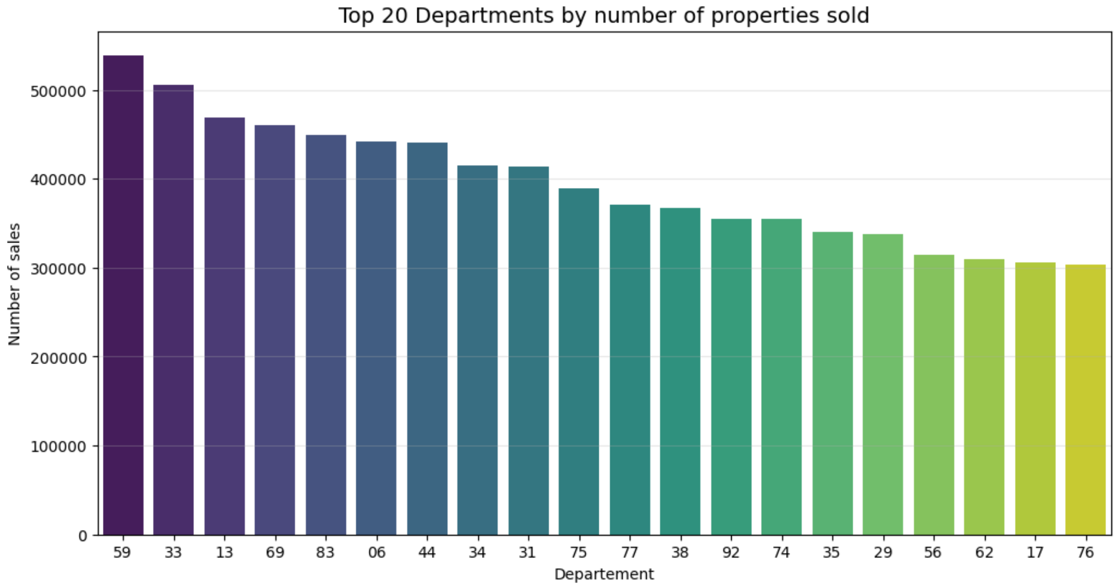
33378

[38]:

	code_departement	nom_commune
0	01	Ceyzériat
13130	01	Ruffieu
12926	01	Échallon
12892	01	Mérignat
12786	01	Armix
12564	01	Boyeux-Saint-Jérôme
12552	01	Géovreisset
12478	01	Villes
12423	01	Conand
12204	01	Plagne

[77]:
```
# Number of properties sold by department
sales_by_dept = (
    df_dvf.groupby('code_departement')
    .size()
    .reset_index(name='num_vendite')
    .sort_values(by='num_vendite', ascending=False)
)

plt.figure(figsize=(12,6))
sns.barplot(
    data=sales_by_dept.head(20),
    x='code_departement', y='num_vendite',
    palette='viridis'
)
plt.title("Top 20 Departments by number of properties sold", fontsize=14)
plt.xlabel('Departement')
plt.ylabel('Number of sales')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



[7]:
```python
# Evolution of average prices over time

# the date must be in datetime format
df_dvf['date_mutation'] = pd.to_datetime(df_dvf['date_mutation'],␣
 ↪errors='coerce')

# Add column Year-Month
df_period['year_month'] = df_period['date_mutation'].dt.to_period('M')

# Calculate average monthly price
monthly_avg = (
    df_period.groupby('year_month')['valeur_fonciere']
    .mean()
    .reset_index()
    .sort_values(by='year_month')
)
monthly_avg['year_month'] = monthly_avg['year_month'].astype(str)

# Line plot to show price's evolution
plt.figure(figsize=(12,6))
sns.lineplot(
    data=monthly_avg,
    x='year_month', y='valeur_fonciere',
    marker='o', color='darkorange'
)
plt.title('Evolution of average monthly prices (2019-2024)', fontsize=14)
plt.xlabel('Month')
```

```
plt.ylabel('Average price')
plt.xticks(rotation=45)
plt.grid(alpha=0.4)
plt.tight_layout()
plt.show()
```

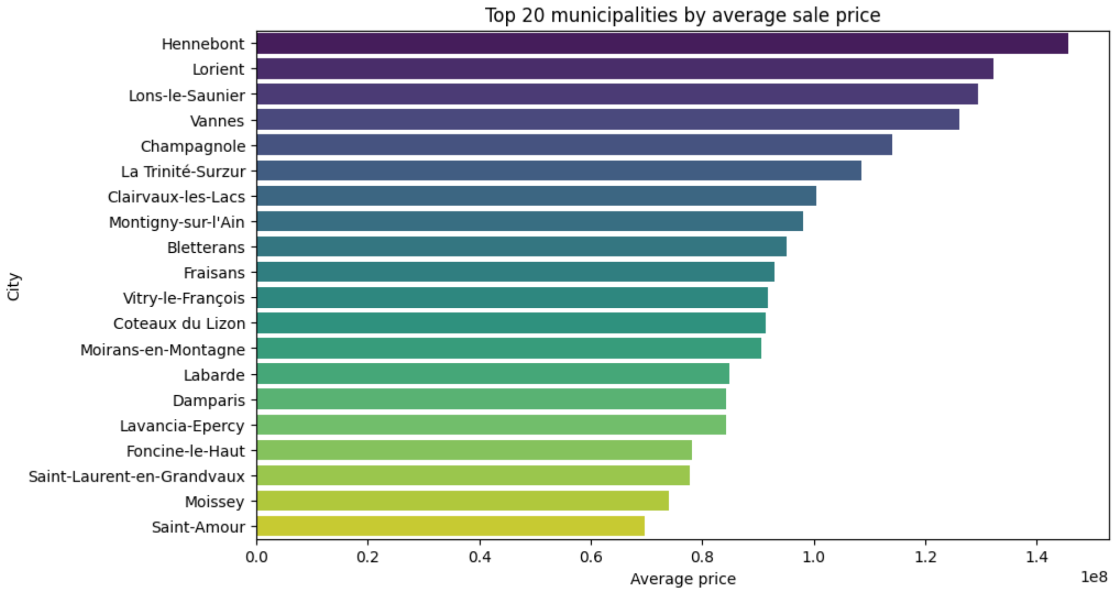
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[7], line 7
      4 df_dvf['date_mutation'] = pd.to_datetime(df_dvf['date_mutation'],␣
 ↪errors='coerce')
      6 # Add column Year-Month
----> 7 df_period['year_month'] = df_period['date_mutation'].dt.to_period('M')
      9 # Calculate average monthly price
     10 monthly_avg = (
     11     df_period.groupby('year_month')['valeur_fonciere']
     12     .mean()
     13     .reset_index()
     14     .sort_values(by='year_month')
     15 )

NameError: name 'df_period' is not defined
```

[78]:

```
# Average prices by city
price_by_city = df_dvf.groupby('nom_commune')['valeur_fonciere'].mean().
 ↪reset_index()
top_cities = price_by_city.sort_values('valeur_fonciere', ascending=False).
 ↪head(20)

plt.figure(figsize=(10,6))
sns.barplot(data=top_cities, y='nom_commune', x='valeur_fonciere',␣
 ↪palette='viridis')
plt.title("Top 20 municipalities by average sale price")
plt.xlabel("Average price")
plt.ylabel("City")
plt.show()
```



2.1.8 Evolution of average monthly prices (2019–2024)

The chart shows how the average property sale price in France has evolved month by month concering the dataset period 2019 and 2024. Despite several sharp spikes likely caused by exceptional transactions or outliers, the overall trend appears relatively stable. This suggests that, while individual high-value sales occasionally distort the monthly average, the broader market did not experience major long-term volatility during this period.

[8]:
```
# Convert the date column to datetime format
df_dvf['date_mutation'] = pd.to_datetime(df_dvf['date_mutation'],␣
↪errors='coerce')

# Filter dataset for the years 2019-2024
df_period = df_dvf[df_dvf['date_mutation'].dt.year.between(2019, 2024)]

# Add Year-Month column for monthly aggregation
df_period['year_month'] = df_period['date_mutation'].dt.to_period('M')

# Calculate average sale price per month
monthly_avg = (
    df_period.groupby('year_month')['valeur_fonciere']
    .mean()
    .reset_index()
    .sort_values(by='year_month')
)

# Convert the period to datetime for proper time formatting
```

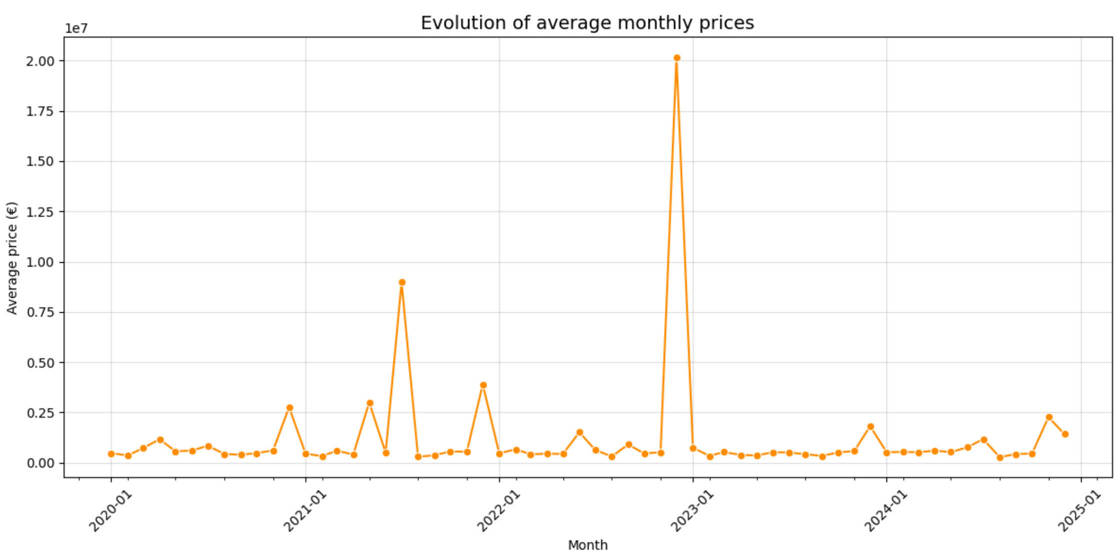
```python
monthly_avg['year_month'] = monthly_avg['year_month'].dt.to_timestamp()

# Line plot showing temporal evolution of average prices
plt.figure(figsize=(12,6))
sns.lineplot(
    data=monthly_avg,
    x='year_month', y='valeur_fonciere',
    marker='o', color='darkorange'
)

plt.title('Evolution of average monthly prices', fontsize=14)
plt.xlabel('Month')
plt.ylabel('Average price (€)')

# X-axis: one major tick per year and minor ticks every 3 months
ax = plt.gca()
ax.xaxis.set_major_locator(mdates.YearLocator())
ax.xaxis.set_minor_locator(mdates.MonthLocator(interval=3))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))

plt.xticks(rotation=45)
plt.grid(alpha=0.4)
plt.tight_layout()
plt.show()
```

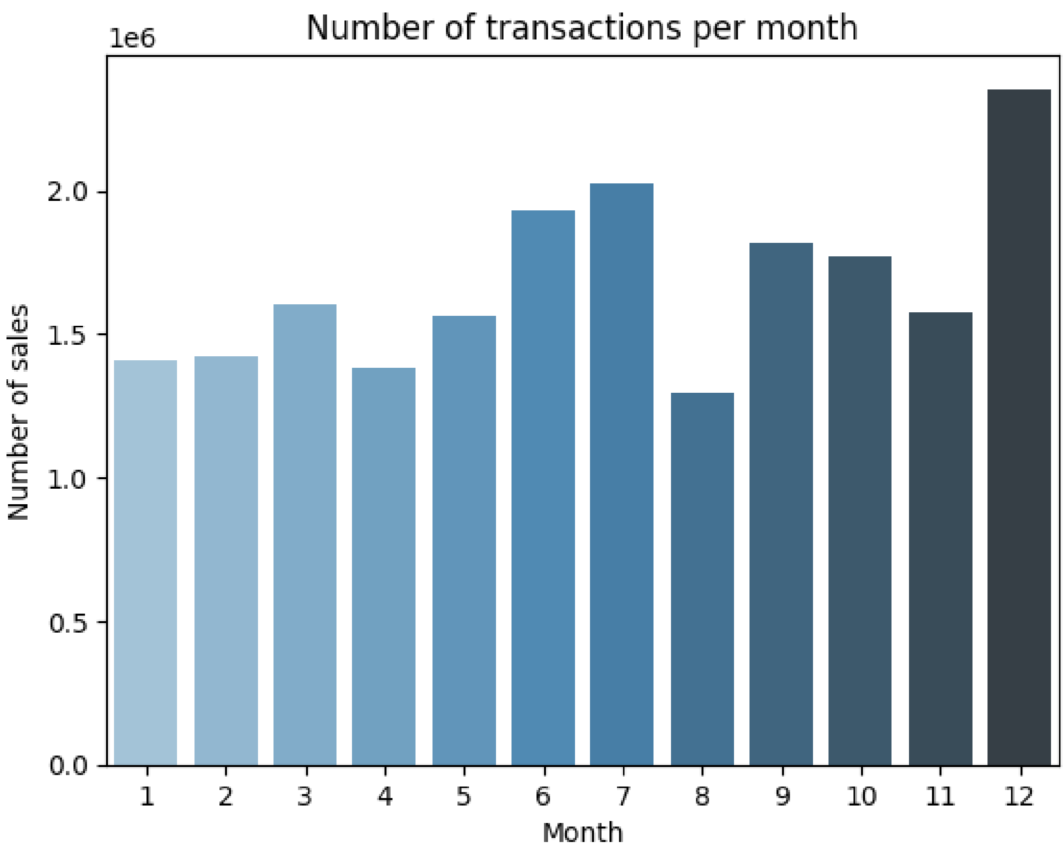


2.1.9 Number of transactions per month

The chart highlights the seasonality of real estate transactions in France. Sales activity tends to increase during the summer months, peaking in June and July, and again in December, likely due to year-end transactions and fiscal timing. Conversely, August shows a notable drop, reflecting the slowdown typical of the vacation period.

[9]:
```
# Extract the month from the transaction date and count how many sales occur in␣
↪each month
df_dvf['month'] = df_dvf['date_mutation'].dt.month
transactions_per_month = df_dvf.groupby('month').size().
↪reset_index(name='count')

# Bar plot showing the number of transactions by month
sns.barplot(
    data=transactions_per_month,
    x='month',
    y='count',
    palette='Blues_d'
)
plt.title("Number of transactions per month")
plt.xlabel("Month")
plt.ylabel("Number of sales")
plt.show()
```



2.2 Conclusion

This preliminary exploration of the DVF dataset provides a general understanding of its structure, data quality, and main patterns before conducting a formal Exploratory Data Analysis (EDA).

2.2.1 1. Dataset Structure and Quality

- The dataset is extremely large (~20 million records, 40 columns) and contains both administrative and transactional variables.
- Several cadastral-related columns show over 90% missing values and can be safely removed.
- The core variables relevant for predictive modeling — `valeur_fonciere`, `surface_reelle_bati`, `nombre_pieces_principales`, `type_local`, `date_mutation`, `longitude`, and `latitude` — are complete and consistent.

2.2.2 2. Property Value Distribution

- Property sale prices are heavily right-skewed, with a few extremely high values.
- A logarithmic transformation (`log10`) normalizes the distribution and improves interpretability.
- This confirms that future models should work on a log-transformed price variable.

2.2.3 3. Property Type Differences

- "Appartements" represent the majority of transactions, while "Maisons" tend to have higher median prices and larger variability.
- Property type is therefore a key categorical predictor to include in the Machine Learning models.

2.2.4 4. Relationship Between Area and Value

- A clear positive but sublinear correlation exists between built area and property value.
- Some small but high-priced properties likely correspond to urban premium locations.
- The price-per-square-meter ratio will be an important engineered feature.

2.2.5 5. Temporal Patterns

- Transaction volumes exhibit strong annual seasonality, with peaks in spring and early summer.
- Average monthly prices increase from 2019 to 2022, then stabilize slightly after 2023.
- Temporal variables should capture both trend and seasonality components.

2.2.6 6. Geographic Distribution

- Sales are concentrated in Île-de-France, the southeast coast, and major urban centers like the capital (Paris).
- Significant spatial heterogeneity is visible in both transaction density and price levels.
- Geographic variables (latitude, longitude, department) play a central role in property cost.

2.2.7 Summary

The DVF dataset provides a rich and reliable foundation for predictive modeling of property sale prices.
Despite some missing administrative fields, the essential features are clean and informative.
The observed temporal, spatial, and structural heterogeneity indicates that future models should combine log-transformed target values, spatial features, and time-based variables to capture the complex dynamics of the French real estate market.