

Projet naiades

NB: Cela peut paraître long, mais nous avons dû télécharger le notebook en PDF depuis Google Colab. Colab utilise par défaut une police d'écriture relativement grande, ce qui explique le nombre de pages élevé dans le PDF généré.

Nous vous fournissons également la version `.ipynb` car elle est plus agréable à lire. Si vous n'avez pas Jupyter Notebook installé, vous pouvez l'ouvrir directement depuis votre navigateur avec Google Colab.

Ce projet s'intéresse à la relation entre les propriétés physico-chimiques de l'eau et son état biologique, avec pour objectif de mieux comprendre les interactions entre ces deux dimensions à partir de données existantes.

Nous avons cherché à déterminer si ces données permettent de révéler des modèles spatiaux pertinents, comme les hydroécorégions, et à explorer comment les différents paramètres influencent l'état des écosystèmes aquatiques.

Pour cela, nous avons appliqué des méthodes de clustering à des données collectées à diverses saisons, tout en considérant un éventuel décalage temporel entre les mesures physico-chimiques et biologiques, afin de refléter la dynamique des écosystèmes. Ces données ont été soigneusement nettoyées, préparées et agrégées par saison, ce qui nous a permis d'identifier des regroupements cohérents d'hydro-eco-stations et d'en analyser les similitudes écologiques.

Par la suite, avec le décalage temporel qui nous paraît le meilleur, nous avons également exploré une approche de régression pour prédire l'état biologique de l'eau (I2M2) à partir des paramètres physico-chimiques, des caractéristiques des stations et de leur temporalité. Cette approche vise à évaluer la capacité de ces facteurs à expliquer et prédire l'état des écosystèmes aquatiques (hydrobiologiques dans ce cas), en tenant compte de la complexité et de la variabilité de ces interactions.

Importation des bibliothèques

```
import numpy as np
import pandas as pd
import geopandas as gpd
import shapely.geometry as geom
from shapely import geometry as geom
from matplotlib.patches import Patch
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as mcolors
```

```

import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
from scipy.stats import zscore
import zstandard as zstd
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import SimpleImputer
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import davies_bouldin_score
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from sklearn.preprocessing import LabelEncoder

```

Chargement des données

```

# stations
df_load_stations =
pd.read_csv('data/stations_hb.csv.zst',sep=',',escapechar = '\\\\')
df_stations = df_load_stations.copy() # copy pour nettoyer etc mais
garder l'original pour recuperer les infos

# pc : physicochimie
f="data/donnees_physicochimie.csv.zst"
pc_sample = pd.read_csv(f,nrows=1)
pc_list_cols = pc_sample.columns
pc_list_cat = pc_list_cols[pc_list_cols.str.startswith((
    'Lb','Nom','Mnemo',
    'Cd','Sym','Com'))]
pc_dict_cat = {col: 'category' for col in pc_list_cat}
df_load_pc = pd.read_csv(
    f,
    sep=',',
    engine='c',
    escapechar='\\\\',
    dtype=pc_dict_cat,
    parse_dates=[7],
    iterator=False)
df_pc = df_load_pc.copy()

# hydrobio
df_load_hydrobio =
pd.read_csv('data/donnees_hydrobio.csv.zst',sep=',',escapechar = '\\\\')
df_hydrobio = df_load_hydrobio.copy()

# hydroecoregion
df_load_hydroregions = gpd.read_file("data/Hydroecoregion1-shp.zip")
df_hydroregions = df_load_hydroregions.copy()

```

Analyse exploratoire

Objectifs

L'objectif principal de cette analyse exploratoire est de comprendre le contenu de chaque jeu de données utilisé dans le projet, d'identifier les colonnes pertinentes ainsi que celles qui peuvent être supprimées, et de déterminer comment remodeler les données pour les rendre exploitables pour l'analyse.

Description des jeux de données

Le projet repose sur plusieurs jeux de données, comprenant des mesures physico-chimiques, des informations hydrobiologiques, des données sur les stations de mesure, et des données géographiques des hydroécorégions.

1. Les **données des stations de mesure** fournissent des informations nécessaires pour localiser chaque station de mesure (latitude et longitude), ainsi que des identifiants uniques permettant de relier les stations entre les différents jeux de données.
2. Les **données physico-chimiques** contiennent les mesures physico-chimiques des eaux (par exemple, nitrates, phosphates, pH...). Chaque mesure est associée à une station, à une date, ainsi qu'à un support et une fraction d'analyse spécifique.
3. Les **données hydrobiologiques** comprennent l'indice écologique I2M2 évaluant la qualité biologique des eaux. Ces indices sont liés aux stations et aux dates de prélèvement.
4. Les **données des hydroécorégions** sont des unités spatiales définies sur la base de critères écologiques similaires, et permettent de classifier les différents milieux aquatiques à travers la France.

Exploration des données géographiques

```
df_stations.head(3)
```

	CdStationMesureEauxSurface	LbStationMesureEauxSurface \
0	01000477	LA SLACK À RINXENT (62)
1	01000602	COLOGNE à BUIRE COURCELLES (80)
2	01000605	L'OMIGNON À DEVISE (80)

	DurStationMesureEauxSurface	CoordXStationMesureEauxSurface \
0	NaN	610228.78
1	NaN	700318.40
2	NaN	700279.85

	CoordYStationMesureEauxSurface	CdProjStationMesureEauxSurface \
0	7078879.90	26
1	6980033.60	26
2	6973284.26	26

	LibelleProjection	CodeCommune	LbCommune
CodeDepartement	... \		
0	RGF93 / Lambert 93	62711	RINXENT
62	...		
1	RGF93 / Lambert 93	80150	BUIRE-COURCELLES
80	...		
2	RGF93 / Lambert 93	80239	DEVISE
80	...		

	DateMAJInfosStationMesureEauxSurface	FinaliteStationMesureEauxSurface	\
0		2015-12-14 00:00:00	
Nan			
1		2015-12-14 00:00:00	
Nan			
2		2015-12-14 00:00:00	
Nan			

	LocPreciseStationMesureEauxSurface	\
0	Lieu-dit Ferme du Château. La Planche du Devin	
1	MOULIN DE BINARD PONT D 194 E	
2	ROUTE DE L'EGLISE	

	CodeNatureStationMesureEauxSurface	LibelleNatureStationMesureEauxSurface	\
0		M	Station de mesure
Manuelle			
1		M	Station de mesure
Manuelle			
2		M	Station de mesure
Manuelle			

	AltitudePointCaracteristique	PkPointTronconEntiteHydroPrincipale	\
0	0.0	980.48	
1	0.0	992.58	
2	0.0	993.69	

	PremierMoisAnneeEtiage	SuperficieBassinVersantReel	\
0	6.0	NaN	
1	6.0	NaN	
2	6.0	NaN	

	SuperficieBassinVersantTopo	
0	0.0	
1	0.0	
2	0.0	

[3 rows x 39 columns]

La table station est utile pour situer les stations ainsi que pour extraire l'identifiant pour joindre les tables sur les stations.

```
df_hydroregions.head(3)

   gid CdHER1    NomHER1
geometry
0     1      16      CORSE  POLYGON ((9.43319 43.00468, 9.4357
42.99999, 9...
1     2      12  ARMORICAIN  POLYGON ((-2.61068 48.55022, -2.61268
48.54898...
2     3      13      LANDES  MULTIPOLYGON (((-1.04228 45.54443, -
1.03836 45...

# On vérifie qu'un code d'hydroécorégion (CdHER1) correspond bien à un
# seul nom d'hydroécorégion (NomHER1)
print(df_hydroregions['CdHER1'].value_counts())

CdHER1
16    1
12    1
10    1
9     1
3     1
17    1
6     1
5     1
19    1
8     1
15    1
4     1
18    1
20    1
7     1
2     1
21    1
11    1
1     1
14    1
13    1
22    1
Name: count, dtype: int64
```

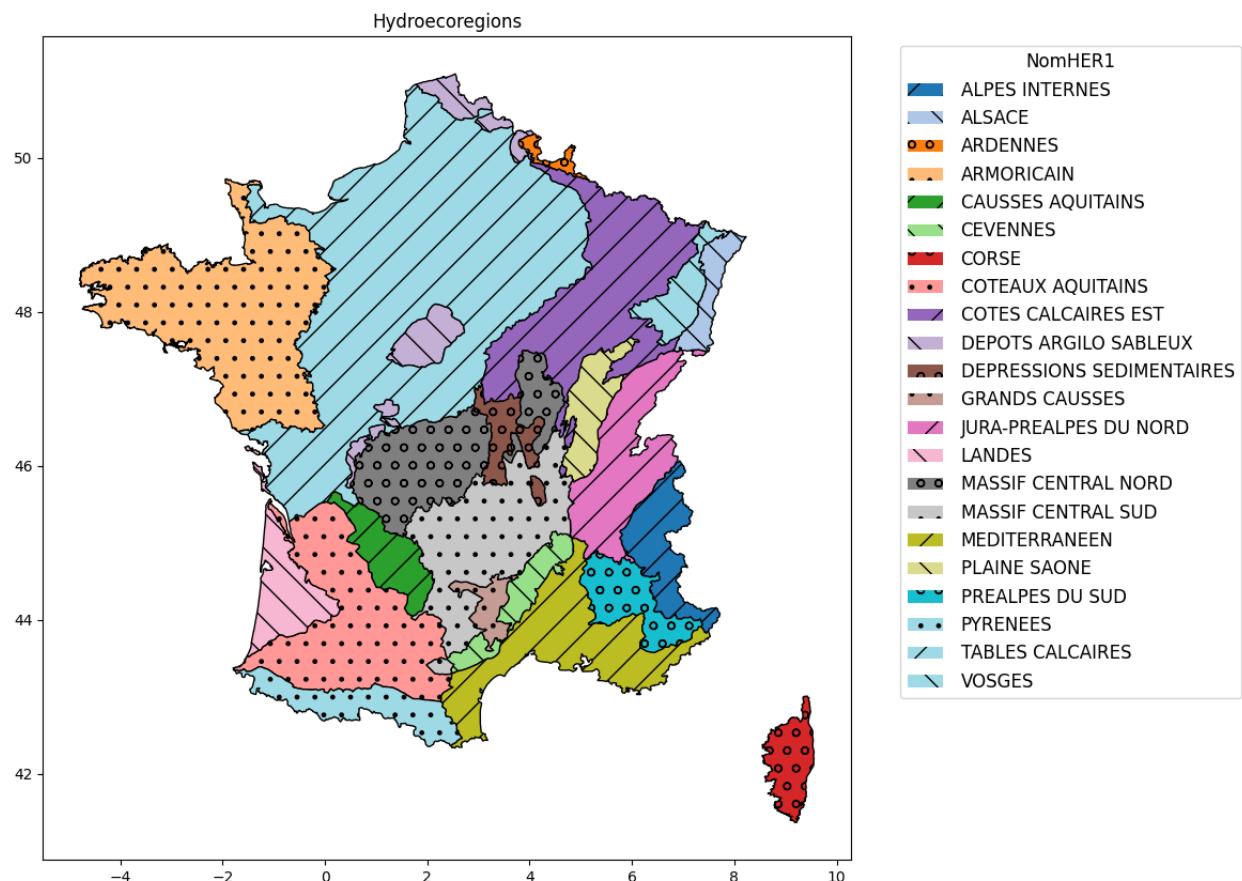
On a bien un CdHER1 pour un NomHER1, donc on va afficher la carte avec NomHER1 car cela est plus parlant pour l'utilisateur.

```
# Affichage des hydroeco regions
fig, ax = plt.subplots(figsize=(20, 10))
colors = plt.colormaps['tab20']
hatches = ['/', '\\\\', 'o', '.']
```

```

legend_elements = []
for i, (name, region) in enumerate(df_hydroregions.groupby('NomHER1')):
    patch = region.plot(ax=ax, color=colors(i), hatch=hatches[i % len(hatches)], edgecolor='black', label=name)
    legend_elements.append(Patch(facecolor=colors(i), hatch=hatches[i % len(hatches)], label=name))
ax.set_title('Hydroecoregions')
ax.legend(handles=legend_elements, title='NomHER1',
bbox_to_anchor=(1.05, 1), loc='upper left', fontsize='large',
title_fontsize='large', handletextpad=1)
plt.show()

```



Nous allons maintenant situer les différentes stations sur la carte des hydroécorégions.

```

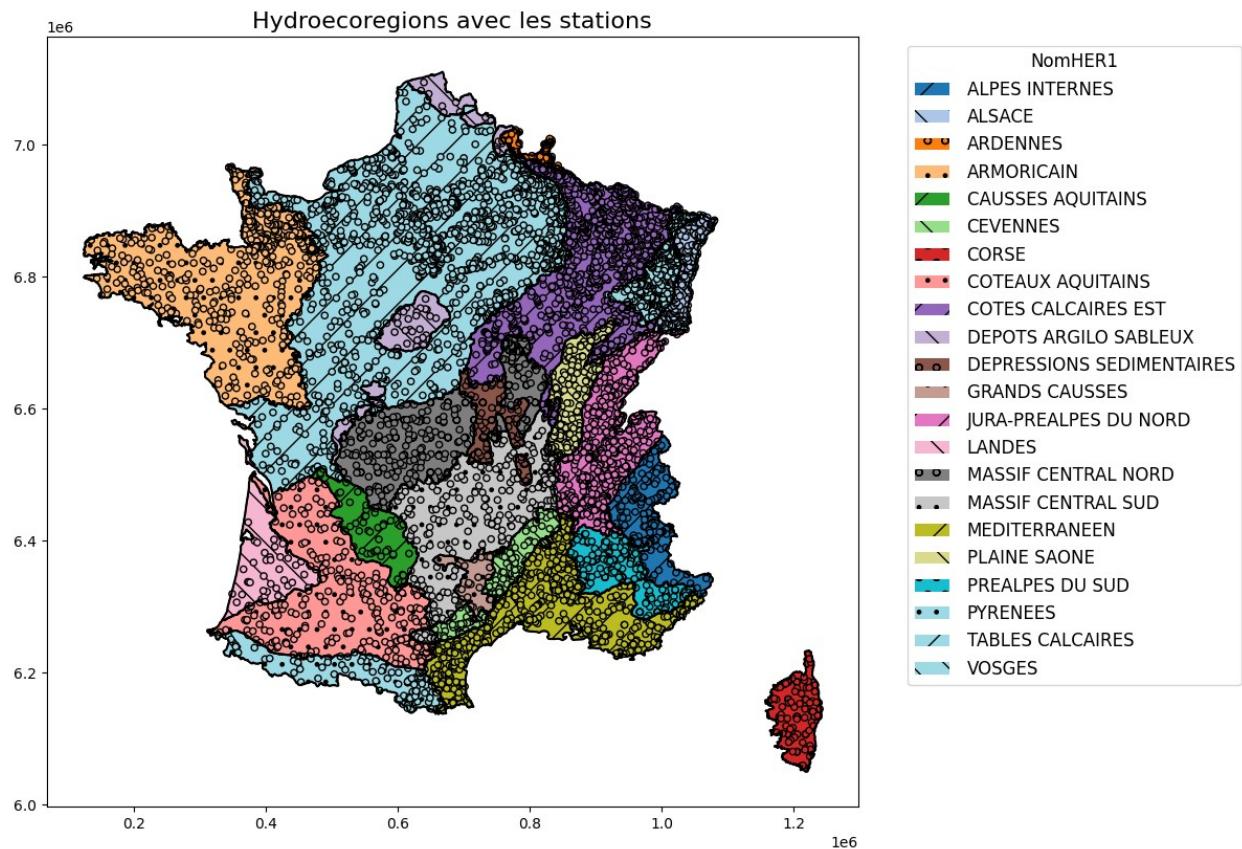
# Affichage des stations sur les hydroecoregions
crs_lambert =
'PROJCS["RGF_1993_Lambert_93",GEOGCS["GCS_RGF_1993",DATUM["D_RGF_1993",
,SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0
,UNIT["Degree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],
PARAMETER["False_Easting",700000.0],PARAMETER["False_Northing",660000.0],
PARAMETER["Central_Meridian",3.0],PARAMETER["Standard_Parallel

```

```

_1",49.0],PARAMETER["Standard_Parallel_2",44.0],PARAMETER["Latitude_of
_Origin",46.5],UNIT["Meter",1.0]]'
x_col = 'CoordXStationMesureEauxSurface'
y_col = 'CoordYStationMesureEauxSurface'
carto_i2m2 = gpd.GeoDataFrame(df_stations,crs=crs_lambert, geometry =
gpd.GeoSeries(df_stations.agg(lambda
x:geom.Point(x.loc[x_col],x.loc[y_col]) ,axis=1)))
HER_stations=carto_i2m2.sjoin(df_hydroregions.to_crs(crs_lambert),pred
icate='within').to_crs(crs_lambert)
fig, ax = plt.subplots(1, 1, figsize=(10, 30))
colors = plt.colormaps['tab20']
hatches = ['/', '\\", 'o', '.']
legend_elements = []
color_mapping = {}
for i, (name, region) in
enumerate(df_hydroregions.groupby('NomHER1')):
    region = region.to_crs(crs_lambert)
    patch = region.plot(ax=ax, color=colors(i), hatch=hatches[i %
len(hatches)], edgecolor='black', label=name)
    legend_elements.append(Patch(facecolor=colors(i), hatch=hatches[i %
len(hatches)], label=name))
    color_mapping[name] = colors(i)
station_colors = HER_stations['NomHER1'].map(color_mapping)
HER_stations.plot(ax=ax, color=station_colors, markersize=20,
edgecolor='black')
HER_lambert = df_hydroregions.to_crs(crs_lambert)
HER_lambert.boundary.plot(ax=ax, color='black')
ax.legend(handles=legend_elements, title='NomHER1',
bbox_to_anchor=(1.05, 1), loc='upper left', fontsize='large',
title_fontsize='large', handletextpad=1)
ax.set_title('Hydroecoregions avec les stations', fontsize=16)
plt.show()

```



Nous observons maintenant la répartition des stations dans les différentes hydroécorégions.

```
stations_par_hydroecoregion =
HER_stations.groupby('NomHER1').size().reset_index(name='nombre_de_stations')
print(stations_par_hydroecoregion)
```

	NomHER1	nombre_de_stations
0	ALPES INTERNES	156
1	ALSACE	352
2	ARDENNES	63
3	ARMORICAIN	445
4	CAUSSES AQUITAINS	47
5	CEVENNES	106
6	CORSE	63
7	COTEAUX AQUITAINS	265
8	COTES CALCAIRES EST	1008
9	DEPOTS ARGILLO SABLEUX	47
10	DEPRESSIONS SEDIMENTAIRES	63
11	GRANDS CAUSSES	25
12	JURA-PREALPES DU NORD	632
13	LANDES	38
14	MASSIF CENTRAL NORD	221
15	MASSIF CENTRAL SUD	264

16	MEDITERRANEEN	574
17	PLAINE SAONE	223
18	PREALPES DU SUD	167
19	PYRENEES	116
20	TABLES CALCAIRES	1360
21	VOSGES	313

```

stations_par_hydroecoregion =
stations_par_hydroecoregion.sort_values(by='nombre_de_stations',
ascending=True)
fig = px.bar(stations_par_hydroecoregion, x='NomHER1',
y='nombre_de_stations', title='Nombre de stations par hydroécorégion',
labels={'NomHER1': 'Hydroécorégion', 'nombre_de_stations': 'Nombre de Stations'})
fig.update_layout(xaxis_tickangle=45, width=900, height=600,
xaxis_title="Hydroécorégion", yaxis_title="Nombre de Stations",
title_font_size=18, xaxis_title_font_size=14,
yaxis_title_font_size=14)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
    {"alignmentgroup": "True", "hovertemplate": "Hydroécorégion=%{x}<br>Nombre de Stations=%{y}</extra></extra>", "legendgroup": "", "marker": {"color": "#636efa", "pattern": {"shape": ""}}, "name": "", "offsetgroup": "", "orientation": "v", "showlegend": false, "textposition": "auto", "type": "bar", "x": ["GRANDS CAUSSES", "LANDES", "CAUSSES AQUITAINS", "DEPOTS ARGILOSABLEUX", "DEPRESSIONS", "SEDIMENTAIRES", "ARDENNES", "CORSE", "CEVENNES", "PYRENEES", "ALPES INTERNES", "PREALPES DU SUD", "MASSIF CENTRAL NORD", "PLAINE SAONE", "MASSIF CENTRAL SUD", "COTEAUX", "AQUITAINS", "VOSGES", "ALSACE", "ARMORICAIN", "MEDITERRANEEN", "JURA-PREALPES DU NORD", "COTES CALCAIRES EST", "TABLES CALCAIRES"], "xaxis": "x", "y": [25, 38, 47, 47, 63, 63, 106, 116, 156, 167, 221, 223, 264, 265, 313, 352, 445, 574, 632, 1008, 1360], "yaxis": "y"}], "layout": {"barmode": "relative", "height": 600, "legend": {"tracegroupgap": 0}, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min": 0, "orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min": 0, "orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, "choropleth": [{"colorbar": {}}]}]
```

```

{"outlineWidth":0,"ticks":[],"type":"choropleth"}], "contour": [{"colorbar":{"outlineWidth":0,"ticks":[],"colorscale": [[0,"#0d0887"],[0.1111111111111111,"#46039f"], [0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"], [0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"], [0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"], [0.8888888888888888,"#fdca26"]], [1,"#f0f921"]}], "type":"contour"}], "contourcarpet": [{"colorbar":{"outlineWidth":0,"ticks":[],"type":"contourcarpet"}}, {"heatmap": [{"colorbar":{"outlineWidth":0,"ticks":[],"colorscale": [[0,"#0d0887"],[0.1111111111111111,"#46039f"], [0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"], [0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"], [0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"], [0.8888888888888888,"#fdca26"]], [1,"#f0f921"]}], "type":"heatmap"}], "heatmapgl": [{"colorbar":{"outlineWidth":0,"ticks":[],"colorscale": [[0,"#0d0887"],[0.1111111111111111,"#46039f"], [0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"], [0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"], [0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"], [0.8888888888888888,"#fdca26"]], [1,"#f0f921"]}], "type":"heatmapgl"}], "histogram": [{"marker": {"pattern": "fillmode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "histogram"}], "histogram2d": [{"colorbar":{"outlineWidth":0,"ticks":[],"colorscale": [[0,"#0d0887"],[0.1111111111111111,"#46039f"], [0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"], [0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"], [0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"], [0.8888888888888888,"#fdca26"]], [1,"#f0f921"]}], "type": "histogram2d"}], "histogram2dcontour": [{"colorbar":{"outlineWidth":0,"ticks":[],"colorscale": [[0,"#0d0887"],[0.1111111111111111,"#46039f"], [0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"], [0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"], [0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"], [0.8888888888888888,"#fdca26"]], [1,"#f0f921"]}], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar":{"outlineWidth":0,"ticks":[],"type": "mesh3d"}}, {"parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"type": "parcoords"}]}, {"pie": [{"automargin": true, "type": "pie"}]}, {"scatter": [{"fillpattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter"}]}, {"scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"type": "scatter3d"}]}, {"scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"type": "scattercarpet"}]}, {"scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"type": "scattergeo"}]}, {"scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": []}}}, {"type": "scattergl"}]}]

```

```

  {"outlinewidth":0,"ticks":""}], "type":"scattergl"}], "scattermapbox": [{"marker":{"colorbar": {"outlinewidth":0,"ticks":""}, "type":"scattermapbox"}], "scatterpolar": [{"marker":{"colorbar": {"outlinewidth":0,"ticks":""}, "type":"scatterpolar"}], "scatterpolargl": [{"marker":{"colorbar": {"outlinewidth":0,"ticks":""}, "type":"scatterpolargl"}], "scatterternary": [{"marker":{"colorbar": {"outlinewidth":0,"ticks":""}, "type":"scatterternary"}], "surface": [{"colorbar":{"outlinewidth":0,"ticks":""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type":"surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}], "layout": {"annotationdefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth":0,"ticks":""}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbcc4"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}], "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FEBC52"}, "font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": 

```

```

{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}, "title": {"font": {"size": 18}, "text": "Nombre de stations par hydroécorégion"}, "width": 900, "xaxis": {"anchor": "y", "domain": [0, 1], "tickangle": 45}, "title": {"font": {"size": 14}, "text": "Hydroécorégion"}}, "yaxis": {"anchor": "x", "domain": [0, 1]}, "title": {"font": {"size": 14}, "text": "Nombre de Stations"}}}}

```

Nous pouvons voir que la répartition des stations n'est pas uniforme, et que le nombre de stations par hydroécorégion peut varier de 25 à 1360. Cela pourrait rendre la caractérisation plus complexe, d'autant plus que nous ne savons pas encore combien de stations seront disponibles après la préparation des données.

Exploration des données physico-chimiques et hydrobiologiques

Physico-chimiques

```

df_pc.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8917443 entries, 0 to 8917442
Data columns (total 49 columns):
 #   Column           Dtype  
 --- 
 0   CdStationMesureEauxSurface  category
 1   LbStationMesureEauxSurface  category
 2   CdSupport                  category
 3   LbSupport                  category
 4   CdFractionAnalysee         category
 5   LbFractionAnalysee         category
 6   CdPrelevement              category
 7   DatePrel                   datetime64[ns]
 8   HeurePrel                  object  
 9   CdParametre                category
 10  LbLongParamètre           category

```

```
11 RsAna          float64
12 CdUniteMesure category
13 SymUniteMesure category
14 CdRqAna        category
15 MnemoRqAna    category
16 CdInsituAna   category
17 LbInsituAna   category
18 ProfondeurPrel float64
19 CdDifficulteAna category
20 MnemoDifficulteAna category
21 LdAna          float64
22 LqAna          float64
23 LsAna          float64
24 IncertAna     float64
25 CdMetFractionnement category
26 NomMetFractionnement category
27 CdMethode      category
28 NomMethode     category
29 RdtExtraction  float64
30 CdMethodeExtraction category
31 NomMethodeExtraction category
32 CdAccreAna    category
33 MnemoAccredAna category
34 AgreAna        float64
35 CdStatutAna   category
36 MnemoStatutAna category
37 CdQualAna     category
38 LbQualAna     category
39 CommentairesAna category
40 ComResultatAna category
41 CdRdd          category
42 NomRdd         category
43 CdProducteur   category
44 NomProducteur  category
45 CdPreleveur    category
46 NomPreleveur   category
47 CdLaboratoire  category
48 NomLaboratoire category
dtypes: category(39), datetime64[ns](1), float64(8), object(1)
memory usage: 1.1+ GB
```

```
df_pc.shape
```

```
(8917443, 49)
```

```
Hydro-biologiques
```

```
df_hydrobio.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43535 entries, 0 to 43534
Data columns (total 21 columns):
 #   Column           Non-Null Count Dtype  
 --- 
 0   Unnamed: 0        43535 non-null  int64  
 1   CdStationMesureEauxSurface 43535 non-null  int64  
 2   LbStationMesureEauxSurface 43535 non-null  object  
 3   CdPointEauxSurf          43115 non-null  float64 
 4   DateDebutOperationPrelBio 43535 non-null  object  
 5   CdSupport              43535 non-null  int64  
 6   LbSupport               43535 non-null  object  
 7   DtProdResultatBiologique 14829 non-null  object  
 8   CdParametreResultatBiologique 43535 non-null  int64  
 9   LbLongParametre         43535 non-null  object  
 10  ResIndiceResultatBiologique 43522 non-null  float64 
 11  CdUniteMesure          43535 non-null  object  
 12  SymUniteMesure         43535 non-null  object  
 13  CdRqIndiceResultatBiologique 43535 non-null  int64  
 14  MnemoRqAna             43535 non-null  object  
 15  CdMethEval             28373 non-null  object  
 16  RefOperationPrelBio    43535 non-null  object  
 17  CdProducteur           43535 non-null  int64  
 18  NomProducteur          43531 non-null  object  
 19  CdAccredRsIndiceResultatBiologique 43343 non-null  float64 
 20  MnAccredRsIndiceResultatBiologique 43343 non-null  object  
dtypes: float64(3), int64(6), object(12)
memory usage: 7.0+ MB

```

`df_hydrobio.shape`

(43535, 21)

Préparation des données physicochimiques

Démarche

Pour notre analyse des hydroécorégions, nous devons exploiter à la fois les données temporelles et spatiales à notre disposition. Notre principal objectif est d'utiliser les données temporelles pour effectuer un clustering, afin de déterminer si nous pouvons retrouver les hydroécorégions existantes à partir des caractéristiques physico-chimiques et biologiques disponibles.

Les données physicochimiques sont composées d'environ de 9 millions de points de mesure, ce qui nécessite une réduction importante pour les rendre exploitables, compte tenu des limites de nos ressources informatiques. Nous avons procédé en 2 étape :

- **Nettoyage des données** : La première étape a été le nettoyage des données. Les duplicitas et les valeurs aberrantes ont été supprimés. Cependant, compte tenu la

quantité de données, nous ne pouvions pas visualiser l'intégralité ni effectuer des analyses sur chaque point individuellement. Nous avons donc mis en place des stratégies pour nettoyer les données sans pour autant perdre des informations significatives.

- **Aggrégation des données** : Nous avons ensuite agrégé les données par stations, par saison et par année, afin de réduire la variabilité et de faciliter l'analyse des tendances. Pour chaque paramètre, nous avons calculé des statistiques résumées telles que la médiane et la moyenne, dans le but de conserver un maximum d'information en réduisant la taille de notre jeu de données. Nous avons également étudié les supports, les fractions, les unités de mesures, et les paramètres mesurés afin de filtrer et ne garder que les informations pertinentes pour l'étude. Et enfin, nous avons remodelé les données afin d'obtenir une table avec pour chaque station, année et saison, des valeurs agrégées pour chaque paramètre physicochimique.

Nettoyage des données

Hypothèse : Nous faisons l'hypothèse que certaines colonnes dans notre jeu de données, telles que le laboratoire, le producteur, le préleveur, et le responsable de la collecte, ne sont pas directement pertinentes pour l'analyse des caractéristiques physicochimiques de l'eau. Ce jeu de données contient essentiellement des métadonnées relatives à la gestion des prélèvements qui ne renseignent pas sur l'état physique ou chimique de l'eau elle-même. De plus, puisque nous prévoyons d'agréger les données par saison et par année, la granularité temporelle de l'heure n'est pas pertinente, nous allons donc éliminer cette colonne.

```
# combien de données dupliquées ?
print(df_pc.duplicated().sum())

11836

# supprimer les doublons
df_pc.drop_duplicates(inplace=True)
```

Caractériser une analyse physico chimique

Comme vu dans le TD, ce qui caractérise une analyse c'est :

- le support
- le paramètre
- la fraction analysée
- l'unité

```
print("Supports :", df_pc['LbSupport'].nunique(), "valeurs ->",
      df_pc['LbSupport'].unique().tolist())
print("Paramètres :", df_pc['LbLongParamètre'].nunique(), "valeurs ->",
      df_pc['LbLongParamètre'].unique().tolist())
print("Fractions :", df_pc['LbFractionAnalysee'].nunique(), "valeurs ->",
      df_pc['LbFractionAnalysee'].unique().tolist())
```

```

print("Unités : ", df_pc['SymUniteMesure'].nunique(), "valeurs ->",
df_pc['SymUniteMesure'].unique().tolist())

Supports : 5 valeurs -> ['Eau', 'Air', 'Sédiments', 'Diatomées
benthiques', 'Gammarides']
Paramètres : 16 valeurs -> ['Matières en suspension', 'Demande
Biochimique en oxygène en 5 jours (D.B.O.5)', "Température de l'Eau",
'Potentiel en Hydrogène (pH)', 'Conductivité à 25°C', 'Oxygène
dissous', 'Taux de saturation en oxygène', 'Phosphore total',
'Turbidité Formazine Néphélosométrique', 'Azote Kjeldahl', 'Diuron',
'Carbone Organique', 'Ammonium', 'Nitrites', 'Nitrates',
'Orthophosphates (P04)']

Fractions : 2 valeurs -> ['Eau brute', "Phase aqueuse de l'eau
(filtrée, centrifugée... )"]

Unités : 15 valeurs -> ['mg/L', 'mg(O2)/L', '°C', 'unité pH', 'µS/cm',
'%', 'mg(P)/L', 'NFU', 'mg(N)/L', 'µg/L', 'mg(C)/L', 'mg(NH4)/L',
'mg(N02)/L', 'mg(N03)/L', 'mg(P04)/L']

```

Pour mieux comprendre nos données physicochimiques, nous allons analyser la distribution des paramètres mesurés ainsi que leurs relations avec les fractions, supports, et unités de mesure. Nous examinerons également les relations entre fractions et supports, afin d'identifier les combinaisons les plus pertinentes et de garantir la cohérence des données pour l'analyse.

Fractions et Supports

```

test = df_pc[['LbSupport','LbFractionAnalysee']].drop_duplicates()

print('Nombre de fractions par support')
test.groupby(['LbSupport']).count()

```

Nombre de fractions par support

	LbFractionAnalysee
LbSupport	
Eau	2
Air	2
Sédiments	2
Diatomées benthiques	1
Gammarides	1

Pour un support donné plusieurs fractions peuvent être analysées.

```

print('Nombre de supports par fraction')
test.groupby(['LbFractionAnalysee']).count()

```

Nombre de supports par fraction

	LbSupport
LbFractionAnalysee	
Eau brute	5
Phase aqueuse de l'eau (filtrée, centrifugée...)	3

Pour une fraction donnée plusieurs supports peuvent être analysés.

Fractions et paramètres

```
test =  
df_pc[['LbLongParamètre', 'LbFractionAnalysee']].drop_duplicates()  
  
print('Nombre de fractions par paramètre')  
test.groupby(['LbLongParamètre']).count()
```

Nombre de fractions par paramètre

	LbFractionAnalysee
LbLongParamètre	
Azote Kjeldahl	1
Conductivité à 25°C	1
Demande Biochimique en oxygène en 5 jours (D.B....)	1
Diuron	1
Matières en suspension	1
Oxygène dissous	1
Phosphore total	1
Potentiel en Hydrogène (pH)	1
Taux de saturation en oxygène	1
Température de l'Eau	1
Turbidité Formazine Néphélométrique	1
Carbone Organique	1
Ammonium	1
Nitrates	1
Nitrites	1
Orthophosphates (P04)	1

Pour un paramètre donné un seul type de fraction est analysé.

```
print('Nombre de paramètres par fraction')  
test.groupby(['LbFractionAnalysee']).count()
```

Nombre de paramètres par fraction

	LbLongParamètre
LbFractionAnalysee	
Eau brute	11
Phase aqueuse de l'eau (filtrée, centrifugée...)	5

Pour une fraction donnée plusieurs paramètres peuvent être analysés.

Supports et paramètres

```
test = df_pc[['LbLongParamètre', 'LbSupport']].drop_duplicates()  
  
print('Nombre de supports par paramètre')  
test.groupby(['LbLongParamètre']).count()
```

Nombre de supports par paramètre

	LbSupport
LbLongParamètre	
Azote Kjeldahl	3
Conductivité à 25°C	4
Demande Biochimique en oxygène en 5 jours (D.B....)	3
Diuron	3
Matières en suspension	3
Oxygène dissous	4
Phosphore total	3
Potentiel en Hydrogène (pH)	5
Taux de saturation en oxygène	4
Température de l'Eau	5
Turbidité Formazine Néphéломétrique	1
Carbone Organique	1
Ammonium	3
Nitrates	3
Nitrites	3
Orthophosphates (P04)	3

Pour un paramètre donné plusieurs supports peuvent être utilisés.

```
print('Nombre de paramètres par support')
test.groupby(['LbSupport']).count()
```

Nombre de paramètres par support

	LbLongParamètre
LbSupport	
Eau	16
Air	14
Sédiments	14
Diatomées benthiques	2
Gammarès	5

Pour un support donné plusieurs paramètres peuvent être analysés.

Unités de mesure et paramètres

```
test = df_pc[['LbLongParamètre', 'SymUniteMesure']].drop_duplicates()

print("Nombre d'unités par paramètre")
test.groupby(['LbLongParamètre']).count()
```

Nombre d'unités par paramètre

	SymUniteMesure
LbLongParamètre	
Azote Kjeldahl	1
Conductivité à 25°C	1

Demande Biochimique en oxygène en 5 jours (D.B....	1
Diuron	1
Matières en suspension	1
Oxygène dissous	1
Phosphore total	1
Potentiel en Hydrogène (pH)	1
Taux de saturation en oxygène	1
Température de l'Eau	1
Turbidité Formazine Néphéломétrique	1
Carbone Organique	1
Ammonium	1
Nitrates	1
Nitrites	1
Orthophosphates (P04)	1

Pour un paramètre donné une seule unité de mesure est utilisée.

```
print("Nombre de paramètres par unité")
test.groupby(['SymUniteMesure']).count()
```

Nombre de paramètres par unité

LbLongParamètre	SymUniteMesure
%	1
NFU	1
mg(N)/L	1
mg(O2)/L	2
mg(P)/L	1
mg/L	1
unité pH	1
°C	1
µS/cm	1
µg/L	1
mg(C)/L	1
mg(NH4)/L	1
mg(NO2)/L	1
mg(NO3)/L	1
mg(P04)/L	1

Pour une unité de mesure donnée plusieurs paramètres peuvent être analysés.

Quelles informations pouvons-nous en extraire ?

Pour chaque paramètre analysé, il n'existe qu'une seule unité de mesure associée. Nous n'avons donc pas besoin de conserver la colonne SymUniteMesure une fois que nous aurons intégré ces données dans notre dataframe pour l'analyse.

D'après les informations fournies dans le cours, un résultat d'analyse (RsAna) se caractérise principalement par :

- Un paramètre physico-chimique (`CdParametre`), comme les nitrates, phosphates, température, pH, ...
- Une unité de mesure (`CdUniteMesure`).
- Une fraction d'analyse (`CdFractionAnalysee`) réalisée à partir d'un support de prélèvement (`CdSupport`).

Étant donné que chaque paramètre est lié à une seule unité de mesure, nous allons pivoter notre table de données afin que chaque résultat d'analyse intègre simultanément le paramètre, le support, et la fraction. Cette organisation facilitera l'analyse et garantira une meilleure cohérence dans nos données.

```
# Pour un paramètre donné, combien y a t il d'unité de mesure, de
# support et de fraction analysee ?
print(df_pc.groupby('LbLongParamètre')
      ['SymUnitéMesure'].nunique().value_counts(), "\n")
print(df_pc.groupby('LbLongParamètre')
      ['LbSupport'].nunique().value_counts(), "\n")
print(df_pc.groupby('LbLongParamètre')
      ['LbFractionAnalysee'].nunique().value_counts())

SymUnitéMesure
1    16
Name: count, dtype: int64

LbSupport
3    9
4    3
5    2
1    2
Name: count, dtype: int64

LbFractionAnalysee
1    16
Name: count, dtype: int64
```

Il n'existe qu'un seul type de fraction analysée pour chaque paramètre.
Pour confirmer cela, nous allons regarder les combinaisons de paramètres, supports et fractions.

```
# Pour un paramètre et un support donné, combien de fraction
# différentes sont analysées ?
grouped = df_pc.groupby(['LbLongParamètre', 'LbSupport'])
['LbFractionAnalysee'].nunique()
multiple_fractions = grouped[grouped > 1]
print(multiple_fractions)

Series([], Name: LbFractionAnalysee, dtype: int64)
```

Il n'y a au maximum qu'une seule fraction par combinaison de paramètre et de support. Cela signifie que la fraction peut être déduite directement à partir du couple (paramètre, support). Nous allons donc faire un regroupement par tuple (**paramètre, support**).

```
params = df_pc.groupby(["LbLongParamètre", 'LbSupport'])

# nombre de valeurs pour chaque tuple
params.size().sort_values(ascending=False)

LbLongParamètre          LbSupport
Potentiel en Hydrogène (pH)    Eau           675940
Température de l'Eau           Eau           675906
Conductivité à 25°C            Eau           668744
Oxygène dissous                Eau           632623
Taux de saturation en oxygène  Eau           613947
                                         ...
Turbidité Formazine Néphélométrique  Diatomées benthiques   0
                                         Gammares           0
Carbone Organique                 Air             0
                                         Sédiments           0
Orthophosphates (P04)            Gammares           0
Length: 80, dtype: int64

# combien de résultats d'analyse pour chaque tuple ?
params_size = params.agg({'RsAna' : ['size']})

# regardons combien de lignes sont vides (RsAna = 0)
params_size[params_size[('RsAna', 'size')] == 0]

RsAna

size
LbLongParamètre          LbSupport
Azote Kjeldahl            Diatomées
benthiques      0           Gammares
0
Conductivité à 25°C       Diatomées
benthiques      0
Demande Biochimique en oxygène en 5 jours (D.B.... Diatomées
benthiques      0           Gammares
0
Diuron                   Diatomées
benthiques      0           Gammares
0
Matières en suspension     Diatomées
benthiques      0
```

```
          Gammares
0
Oxygène dissous Diatomées
benthiques 0
Phosphore total Diatomées
benthiques 0
                                Gammares
0
Taux de saturation en oxygène Diatomées
benthiques 0
Turbidité Formazine Néphélométrique Air
0
                                Séiments
0
                                Diatomées
benthiques 0
                                Gammares
0
Carbone Organique Air
0
                                Séiments
0
                                Diatomées
benthiques 0
                                Gammares
0
                                Diatomées
benthiques 0
                                Gammares
0
Nitrates Diatomées
benthiques 0
                                Gammares
0
Nitrites Diatomées
benthiques 0
                                Gammares
0
Orthophosphates (P04) Diatomées
benthiques 0
                                Gammares
0
params_size[params_size[('RsAna','size')]]!
=0].sort_values(('RsAna','size'),ascending=True)
```

RsAna

size

LbLongParamètre	LbSupport
Température de l'Eau	Gammares
1	
Conductivité à 25°C	Gammares
1	
Oxygène dissous	Gammares
1	
Taux de saturation en oxygène	Gammares
1	
Potentiel en Hydrogène (pH)	Gammares
1	
benthiques 28	Diatomées
Température de l'Eau	Diatomées
benthiques 28	
Diuron	Sédiments
35	
Azote Kjeldahl	Sédiments
60	
Potentiel en Hydrogène (pH)	Sédiments
78	
Phosphore total	Sédiments
78	
Température de l'Eau	Sédiments
78	
Ammonium	Sédiments
78	
Nitrates	Sédiments
78	
Nitrites	Sédiments
78	
Taux de saturation en oxygène	Sédiments
78	
Oxygène dissous	Sédiments
78	
Orthophosphates (P04)	Sédiments
78	
Matières en suspension	Sédiments
78	
Demande Biochimique en oxygène en 5 jours (D.B....	Sédiments
78	
Conductivité à 25°C	Sédiments
78	
Diuron	Air
1046	
Azote Kjeldahl	Air
2136	
Ammonium	Air

3059		
Phosphore total		Air
3061		
Demande Biochimique en oxygène en 5 jours (D.B....)		Air
3145		
Matières en suspension		Air
3148		
Nitrates		Air
3149		
Nitrites		Air
3149		
Orthophosphates (P04)		Air
3149		
Taux de saturation en oxygène		Air
3169		
Oxygène dissous		Air
3395		
Température de l'Eau		Air
3402		
Potentiel en Hydrogène (pH)		Air
3428		
Conductivité à 25°C		Air
3432		
Diuron		Eau
279124		
Turbidité Formazine Néphéломétrique		Eau
420400		
Ammonium		Eau
509158		
Azote Kjeldahl		Eau
525943		
Nitrites		Eau
531995		
Carbone Organique		Eau
535869		
Demande Biochimique en oxygène en 5 jours (D.B....)		Eau
542836		
Orthophosphates (P04)		Eau
543972		
Nitrates		Eau
550019		
Phosphore total		Eau
569020		
Matières en suspension		Eau
587151		
Taux de saturation en oxygène		Eau
613947		
Oxygène dissous		Eau
632623		

Conductivité à 25°C	Eau
668744	
Température de l'Eau	Eau
675906	
Potentiel en Hydrogène (pH)	Eau
675940	

En examinant les mesures, nous pouvons voir que certains supports sont très faiblement représentés : une seule mesure pour les Support Gammes, 28 pour les diatomées benthiques, et maximum 78 pour les sédiments.

En comparaison avec les milliers de mesures pour l'air, et les centaines de milliers de mesures pour l'eau, nous risquons une sous représentation de ces 3 différents supports.

Pour gérer ce déséquilibre, il existe plusieurs stratégies :

- le **suréchantillonnage (over sampling)** pour augmenter artificiellement la présence des supports sous-représentés afin d'équilibrer les données.
- la **suppression des supports faiblement représentés** (ce qu'on a décidé de réaliser : plus tard dans l'analyse, nous supprimerons les supports.)

```
# Conversion en chaînes de caractères de notre colonne caractérisant
# les paramètres (paramètre - support)
param_series = df_pc['LbLongParamètre'].astype(str) + ' - ' +
df_pc['LbSupport'].astype(str)
```

Traiter les seuils (de quantification, de saturation, de détection...)

```
df_pc[['CdRqAna', 'MnemoRqAna']].value_counts()

CdRqAna  MnemoRqAna

1      Résultat > seuil de quantification et < au seuil de
saturation ou Résultat = 0    7764758
10     Résultat < au seuil de quantification
1125177
0      Analyse non faite
10400
2      Résultat < seuil de détection
3157
7      Traces (< seuil de quantification et > seuil de détection)
1812
3      Résultat > seuil de saturation
237
8      Dénombrement > Valeur
64
9      Dénombrement < Valeur
1
11     Echelle Absente
1
Name: count, dtype: int64
```

d'après la doc :

- 1 = ok : rien besoin de faire
- 0 : analyse non faite -> le résultat doit être vide : nous on va supprimer ces lignes
- 2 : le résultat prend alors la valeur du seuil de détection ou du seuil de quantification suivant qu'il est inférieur à l'un de ces deux seuils
- 3 : le résultat donne alors la valeur du seuil de saturation
- 7 : le résultat prend alors la valeur du seuil de détection ou du seuil de quantification suivant qu'il est inférieur à l'un de ces deux seuils. (comme 2)
- 8 : Les codes remarque 8 et 9 doivent être utilisés pour qualifier des résultats fournis par des méthodes de type qualitatif, décrits par rapport à un seuil bien que compris dans la plage d'utilisation courante des méthodes (supérieur au seuil de quantification et inférieur au seuil de saturation).
- 9 : Les codes remarque 8 et 9 doivent être utilisés pour qualifier des résultats fournis par des méthodes de type qualitatif, décrits par rapport à un seuil bien que compris dans la plage d'utilisation courante des méthodes (supérieur au seuil de quantification et inférieur au seuil de saturation).
- 10 : Le résultat quant à lui prend la valeur du seuil de quantification.
- 11 : pas de doc -> supprimer cette valeur

```
print("Avant suppression:", df_pc.shape)
df_pc = df_pc[~df_pc['CdRqAna'].isin(['0', '8', '9', '11'])]
print("Après suppression:", df_pc.shape)
print(df_pc[['CdRqAna', 'MnemoRqAna']].value_counts())

Avant suppression: (8905607, 49)
Après suppression: (8895141, 49)
CdRqAna  MnemoRqAna

1      Résultat > seuil de quantification et < au seuil de
saturation ou Résultat = 0    7764758
10     Résultat < au seuil de quantification
1125177
2      Résultat < seuil de détection
3157
7      Traces (< seuil de quantification et > seuil de détection)
1812
3      Résultat > seuil de saturation
237
Name: count, dtype: int64
```

Nous créons maintenant le dataframe `df_pc_light`, contenant seulement les colonnes pertinentes pour associer un résultat d'analyse à une station, une date de prélèvement et un paramètre.

```
df_pc_light = df_pc[['CdStationMesureEauxSurface', 'DatePrel', 'RsAna']]
df_pc_light['param'] = param_series
```

```
/var/folders/p1/4yqk4cyx3058m3j04rtkr56m0000gn/T/
ipykernel_53919/3914632016.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

```
CdStationMesureEauxSurface    DatePrel      RsAna  \
0                      05005600 2005-07-06     3.000
1                      05200115 2005-09-28     0.700
2                      05001800 2005-01-19     8.100
3                      05001800 2005-01-19     7.800
4                      05001800 2005-01-19   845.000
...
8917438                  ...       ...
8917439                  ...       ...
8917440                  ...       ...
8917441                  ...       ...
8917442                  ...       ...
```

```
                           param
0          Matières en suspension - Eau
1  Demande Biochimique en oxygène en 5 jours (D.B.O5) - Eau
2          Température de l'Eau - Eau
3          Potentiel en Hydrogène (pH) - Eau
4          Conductivité à 25°C - Eau
...
8917438                  ...       ...
8917439                  ...       ...
8917440                  ...       ...
8917441                  ...       ...
8917442                  ...       ...
```

[8895141 rows x 4 columns]

supprimer les lignes où pour un paramètre donné on a moins de 1000 valeurs (comme vu précédemment)

```
df_pc_light = df_pc_light.groupby('param').filter(lambda x:
x['RsAna'].count() > 1000)
```

afficher les lignes où RsAna est vide

```
df_pc_light[df_pc_light['RsAna'].isnull()]
```

Empty DataFrame

Columns: [CdStationMesureEauxSurface, DatePrel, RsAna, param]

Index: []

```

# afficher les doublons
df_pc_light[df_pc_light.duplicated()]

      CdStationMesureEauxSurface DatePrel RsAna \
21474                      05215100 2005-07-04  16.3
35195                      02018000 2005-05-03  15.5
35196                      02018000 2005-05-03   7.9
35199                      02018000 2005-05-03   9.0
35581                      02025500 2005-05-09  11.3
...
8878024                      ...     2022-11-15  13.0
8878932                      ...     2022-11-15 483.0
8878933                      ...     2022-11-15   9.4
8878934                      ...     2022-11-15   8.0
8878936                      ...     2022-11-15  11.9

                           param
21474    Température de l'Eau - Eau
35195    Température de l'Eau - Eau
35196  Potentiel en Hydrogène (pH) - Eau
35199    Oxygène dissous - Eau
35581    Température de l'Eau - Eau
...
8878024    Température de l'Eau - Eau
8878932    Conductivité à 25°C - Eau
8878933    Oxygène dissous - Eau
8878934  Potentiel en Hydrogène (pH) - Eau
8878936    Température de l'Eau - Eau

[83424 rows x 4 columns]

# supprimer les doublons (de nouveau, on avait déjà supprimé les
# doublons mais avant de grouper par paramètre et de ne garder que les
# paramètres avec plus de 1000 valeurs)
print("Avant suppression:", df_pc_light.shape)
df_pc_light.drop_duplicates(inplace=True)
print("Après suppression:", df_pc_light.shape)

Avant suppression: (8894049, 4)
Après suppression: (8810625, 4)

# pour un paramètre donné à une station donné à une date donnée, y a t
il plusieurs résultats d'analyse ?
duplicates =
df_pc_light[df_pc_light.duplicated(subset=['CdStationMesureEauxSurface',
                                             'param', 'DatePrel'], keep=False)]
duplicates.shape

(40987, 4)

```

Il est possible d'avoir plusieurs résultats d'analyse pour un même paramètre à une station donnée, à une date donnée. Cette situation se produit dans environ 41 000 cas. Plutôt que de supprimer ces doublons potentiels, nous choisissons de les conserver, car nous allons agréger les données par date et par station par la suite.

Traiter les valeurs aberrantes dans les données physicochimiques

Lors de l'exploration des données physicochimiques, nous avons constaté la présence de valeurs aberrantes pour certains paramètres, par exemple, une température de l'eau enregistrée à -9999.0. Compte tenu du volume important des données, afficher tous les outliers s'est avéré trop lourd et inefficace. Cependant, il est clair que ces cas doivent être traités pour garantir la qualité de l'analyse.

Nous avons étudié plusieurs manières pour identifier et traiter ces valeurs aberrantes :

- La méthode de l'intervalle interquartile (IQR)
- Le z-score avec différents seuils

La méthode retenue est le z-score avec un seuil de 3.

```
df_pc_cleaned = df_pc_light.copy()
outliers_summary_before = []
for param in df_pc_cleaned['param'].unique():
    df_param = df_pc_cleaned[df_pc_cleaned['param'] == param].copy()
    df_param['zscore'] = zscore(df_param['RsAna'])
    outliers_before = df_param[abs(df_param['zscore']) > 3]
    outliers_summary_before.append({
        "param": param,
        "total_values": len(df_param),
        "outliers_count_before": len(outliers_before),
        "outliers_percent_before": len(outliers_before) /
len(df_param) * 100,
    })
# Imputation des outliers avec la médiane
median_value = df_param['RsAna'].median()
df_param.loc[abs(df_param['zscore']) > 3, 'RsAna'] = median_value
df_pc_cleaned.loc[df_pc_cleaned['param'] == param, 'RsAna'] =
df_param['RsAna']
outliers_summary_before_df = pd.DataFrame(outliers_summary_before)
print("Résumé des outliers avant imputation :")
print(outliers_summary_before_df)
```

Résumé des outliers avant imputation :

		param	total_values	\
0	Matières en suspension	- Eau	584610	
1	Demande Biochimique en oxygène en 5 jours (D.B...)		541160	
2	Température de l'Eau	- Eau	656418	
3	Potentiel en Hydrogène (pH)	- Eau	652628	
4	Conductivité à 25°C	- Eau	649121	
5	Oxygène dissous	- Eau	629463	
6	Taux de saturation en oxygène	- Eau	610779	

7		Phosphore total	- Eau	565122
8	Turbidité Formazine	Néphéломétrique	- Eau	419605
9		Azote Kjeldahl	- Eau	524334
10		Diuron	- Eau	278229
11		Carbone Organique	- Eau	534038
12		Ammonium	- Eau	507401
13		Nitrites	- Eau	530105
14		Nitrates	- Eau	545844
15		Orthophosphates (P04)	- Eau	539900
16		Ammonium	- Air	3059
17		Azote Kjeldahl	- Air	2136
18	Taux de saturation en oxygène		- Air	3169
19		Oxygène dissous	- Air	3395
20		Température de l'Eau	- Air	3402
21		Potentiel en Hydrogène (pH)	- Air	3428
22		Conductivité à 25°C	- Air	3432
23		Matières en suspension	- Air	3148
24	Demande Biochimique en oxygène en 5 jours (D.B...)			3145
25		Nitrites	- Air	3149
26		Nitrates	- Air	3149
27		Orthophosphates (P04)	- Air	3149
28		Phosphore total	- Air	3061
29		Diuron	- Air	1046

	outliers_count_before	outliers_percent_before
0	1460	0.249739
1	535	0.098862
2	4	0.000609
3	386	0.059145
4	3651	0.562453
5	100	0.015887
6	11424	1.870398
7	2275	0.402568
8	801	0.190894
9	3126	0.596185
10	605	0.217447
11	3287	0.615499
12	1513	0.298186
13	3338	0.629687
14	5782	1.059277
15	3855	0.714021
16	47	1.536450
17	40	1.872659
18	50	1.577785
19	27	0.795287
20	2	0.058789
21	33	0.962660
22	36	1.048951
23	46	1.461245

24	51	1.621622
25	61	1.937123
26	23	0.730391
27	19	0.603366
28	14	0.457367
29	24	2.294455

Pour chaque paramètre, nous avons calculé les z-scores des résultats d'analyse et identifié les valeurs dont le z-score dépassait 3. Les outliers identifiés ont été remplacés par la médiane des valeurs du paramètre correspondant.

Moins de 2% des données ont été identifiées comme des outliers pour la majorité des paramètres, nous pensons donc avoir limité la perte d'information.

Seul un paramètre, le Diuron - Air, a présenté un taux d'outliers légèrement supérieur à 2% (2,29%). Ce paramètre est également celui avec le moins de valeurs (environ 1 000).

Nous estimons que cette approche permet de conserver la fiabilité des données tout en traitant efficacement les valeurs aberrantes.

Aggrégation des données physicochimiques par saison

Première visualisation des données agrégées

```
df_pc_season = df_pc_cleaned.copy()
df_pc_season['année'] = df_pc_season['DatePrel'].dt.year +
(df_pc_season['DatePrel'].dt.month == 12)
df_pc_season['saison'] = df_pc_season['DatePrel'].dt.month.map({
    12: 'Hiver', 1: 'Hiver', 2: 'Hiver',
    3: 'Printemps', 4: 'Printemps', 5: 'Printemps',
    6: 'Été', 7: 'Été', 8: 'Été',
    9: 'Automne', 10: 'Automne', 11: 'Automne'
})
df_counts = df_pc_season.groupby(['année',
'saison']).size().reset_index(name='nombre de résultats d\'analyse par
saison')

df_counts
```

	année	saison	nombre de résultats d'analyse par saison
0	2005	Automne	56550
1	2005	Hiver	27418
2	2005	Printemps	47644
3	2005	Été	56389
4	2006	Automne	62823
..
68	2022	Automne	59629
69	2022	Hiver	98052
70	2022	Printemps	82245
71	2022	Été	65427
72	2023	Hiver	19532

```
[73 rows x 3 columns]

season_colors = {
    "Printemps": "#77DD77",
    "Été": "#FFB347",
    "Automne": "#FF6961",
    "Hiver": "#AEC6CF"
}

fig = px.bar(df_counts,
              x='année',
              y='nombre de résultats d\'analyse par saison',
              color='saison',
              barmode='stack',
              labels={'Année': 'année', 'Nombre de valeurs': 'Nombre de Mesures'},
              title='Nombre de Mesures par Année et Saison',
              color_discrete_map=season_colors
)
fig.update_layout(
    xaxis_title='année',
    yaxis_title='nombre de résultats d\'analyse',
    legend_title='saison'
)
fig.show()

{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"alignmentgroup": "True", "hovertemplate": "saison=Automne<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra>", "legendgroup": "Automne", "marker": {"color": "#FF6961", "pattern": [{"shape": ""}]}, "name": "Automne", "offsetgroup": "Automne", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [56550, 62823, 91574, 100844, 100593, 135430, 134633, 154583, 157221, 165068, 168285, 157239, 157854, 164339, 152239, 161915, 148978, 59629], "yaxis": "y"}, {"alignmentgroup": "True", "hovertemplate": "saison=Hiver<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra>", "legendgroup": "Hiver", "marker": {"color": "#AEC6CF", "pattern": [{"shape": ""}]}, "name": "Hiver", "offsetgroup": "Hiver", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], "xaxis": "x", "y": [27418, 45603, 66858, 75682, 80782, 79178, 110390, 110996, 126937, 122334, 147091, 142076, 144503, 141918, 128944, 131669, 130064, 98052, 19532], "yaxis": "y"}, {"alignmentgroup": "True", "hovertemplate": "saison=Printemps<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra>", "legendgroup": "Printemps", "marker": {"color": "#77DD77", "pattern": [{"shape": ""}]}, "name": "Printemps", "offsetgroup": "Printemps", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023], "xaxis": "x", "y": [144503, 141918, 128944, 131669, 130064, 98052, 19532], "yaxis": "y"}]}]
```

```

saison=%{y}<extra></extra> , "legendgroup": "Printemps", "marker": {
  "color": "#77DD77", "pattern": {
    "shape": ""}}, "name": "Printemps", "offsetgroup": "Printemps", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [47644, 57310, 87389, 74211, 98111, 103021, 134445, 155880, 152906, 150270, 170117, 161132, 163524, 172154, 161344, 106454, 149943, 82245], "yaxis": "y"}, {"alignmentgroup": "True", "hovertemplate": "saison=Été<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}<extra></extra> , "legendgroup": "Été", "marker": {
  "color": "#FFB347", "pattern": {
    "shape": ""}}, "name": "Été", "offsetgroup": "Été", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [56389, 64921, 88216, 84607, 95411, 110372, 131427, 158775, 159868, 165810, 169689, 164511, 160539, 174831, 152364, 164193, 155351, 65427], "yaxis": "y"}], "layout": {"barmode": "stack", "legend": {"title": {"text": "saison"}, "tracegroupgap": 0}, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "contour"}], "contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmapgl"}]}], "xaxis": {"dtick": 1, "label": "Saison", "title": "Saison"}, "yaxis": {"dtick": 1, "label": "Nombre de résultats", "title": "Nombre de résultats"}}}
```

```

[0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"],
[0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"],
[0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"],
[0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"],
[1, "#f0f921]], "type": "heatmapgl"}, "histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}],
"histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "histogram2d"}, "histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}, "parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"automargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter3d"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}}, {"colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}}, "header": {"fill": "
```

```

{"color": "#C8D4E3"}, "line": {"color": "white"}], "type": "table"}], "layout": {"annotationdefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbcc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}, "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECB52"}, "font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlake": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}], "title": {}

```

```
{"text": "Nombre de Mesures par Année et Saison", "xaxis": {"anchor": "y", "domain": [0, 1], "title": {"text": "année"}}, "yaxis": {"anchor": "x", "domain": [0, 1], "title": {"text": "nombre de résultats d'analyse"}}}}
```

Nous avons plus de 300 000 résultats par an à partir de 2007 jusqu'en 2021, avec une répartition relativement équilibrée entre les saisons. Pour les années 2005, 2006 et 2022, le nombre de résultats est plus faible, autour de 200 000 résultats par an.

Pour simplifier l'analyse tout en conservant les tendances, nous avons choisi d'agréger les données par station, année et saison.

Pour chaque combinaison de station, année, saison, et paramètre (paramètre - support), nous calculons des statistiques représentatives : médiane et moyenne.

Nous créons un nouveau DataFrame composé des valeurs agrégées. Chaque ligne représentera une station, une année, une saison, et contiendra les valeurs médianes et moyennes pour chaque paramètre mesuré dans les colonnes `median_RsAna` et `mean_RsAna`.

Cela permet de réduire la complexité des données tout en conservant leur représentativité.

Aggrégation par station, année et saison

```
df_pc_season.rename(columns={'CdStationMesureEauxSurface': 'station',
'DatePrel': 'date'}, inplace=True)

agg_functions = { 'RsAna': ['mean', 'median'], }
df_aggregated = df_pc_season.groupby(['station', 'param', 'saison',
'année']).agg(agg_functions).reset_index()
df_aggregated.columns = ['station', 'param', 'saison', 'année',
'mean_RsAna', 'median_RsAna']
print(f"Taille du DataFrame après l'agrégation avec différentes fonctions : {df_aggregated.shape[0]}")
```

Taille du DataFrame après l'agrégation avec différentes fonctions :
20086800

```
df_aggregated.isnull().sum()

station          0
param            0
saison           0
année            0
mean_RsAna     15972975
median_RsAna    15972975
dtype: int64
```

Après l'agrégation, le nombre total de valeurs a augmenté de manière significative (environ 14 millions de lignes). Cela suggère qu'il y a de nombreuses valeurs nulles, et nous allons les supprimer.

```
# supprimer toutes les lignes avec des valeurs nulles
df_aggregated.dropna(inplace=True)
df_aggregated.shape

(4113825, 6)

# calculer combien de données en moins ça fait par rapport à avant
aggregation
print("Nombre de valeurs en moins par rapport à avant aggrégation :",
df_pc_season.shape[0] - df_aggregated.shape[0])
print("Ce qui représente une perte de :", (df_pc_season.shape[0] -
df_aggregated.shape[0]) / df_pc_season.shape[0] * 100, "%")

Nombre de valeurs en moins par rapport à avant aggrégation : 4696800
Ce qui représente une perte de : 53.308363481591826 %
```

Nous avons alors 4 millions de lignes, ce qui est bien plus gérable pour réaliser nos analyses.

Création du dataframe agrégé par saison des données physicochimiques

Nous effectuons une nouvelle transformation du dataframe pour obtenir un résumé agrégé des résultats. L'objectif est de calculer, pour chaque station et chaque saison, les statistiques représentatives des paramètres physicochimiques mesurés.

Nous créons 2 dataframes distincts :

- un contenant les moyennes des résultats d'analyse pour chaque paramètre
 - un autre contenant les médianes des résultats d'analyse pour chaque paramètre

Ce pivotement permet de structurer les données de manière claire et synthétique, facilitant ainsi les analyses ultérieures.

```
params = df_aggregated['param'].unique()

df_pc_agg_saison_median = df_aggregated[['station', 'année', 'saison',
    'param', 'median_RsAna']]
df_pc_agg_saison_mean = df_aggregated[['station', 'année', 'saison',
    'param', 'mean_RsAna']]
# Pivot des données : station, année, saison en index, param en
# colonnes
df_pc_pivot_saison_median =
df_aggregated.pivot_table(index=['station', 'année', 'saison'],
                           columns='param',
                           values='median_RsAna',
                           aggfunc='median')
df_pc_pivot_saison_mean = df_aggregated.pivot_table(index=['station',
    'année', 'saison'],
                           columns='param',
                           values='mean_RsAna',
                           aggfunc='mean')
```

```
df_pc_pivot_saison_median.reset_index(inplace=True)
df_pc_pivot_saison_mean.reset_index(inplace=True)
```

Analyse des données agrégées avec la médiane

```
df_pc_pivot_saison_median.head(5)
```

```
param station année saison Ammonium - Air Ammonium - Eau \
0 05001800 2005 Automne NaN NaN
1 05001800 2005 Hiver NaN NaN
2 05001800 2005 Printemps NaN NaN
3 05001800 2005 Été NaN NaN
4 05001800 2007 Automne NaN 0.025
```

```
param Azote Kjeldahl - Air Azote Kjeldahl - Eau Carbone Organique -
Eau \
0 NaN NaN
NaN
1 NaN NaN
NaN
2 NaN NaN
NaN
3 NaN NaN
NaN
4 NaN 1.0
2.75
```

```
param Conductivité à 25°C - Air Conductivité à 25°C - Eau ... \
0 NaN 603.0 ...
1 NaN 825.0 ...
2 NaN 779.0 ...
3 NaN 630.0 ...
4 NaN 770.0 ...
```

```
param Oxygène dissous - Eau Phosphore total - Air Phosphore total -
Eau \
0 7.50 NaN
0.05
1 11.95 NaN
0.05
2 10.10 NaN
0.05
3 7.90 NaN
0.05
4 9.10 NaN
0.05
```

```
param Potentiel en Hydrogène (pH) - Air Potentiel en Hydrogène (pH) -
Eau \
0 NaN
```

```
7.5
1                               NaN
7.9
2                               NaN
8.0
3                               NaN
7.7
4                               NaN
8.0

param Taux de saturation en oxygène - Air \
0                               NaN
1                               NaN
2                               NaN
3                               NaN
4                               NaN

param Taux de saturation en oxygène - Eau Température de l'Eau - Air \
0                               82.0           NaN
1                               103.5          NaN
2                               104.0          NaN
3                               88.0           NaN
4                               94.0           NaN

param Température de l'Eau - Eau Turbidité Formazine Néphéломétrique
- Eau
0                               19.60
NaN
1                               8.95
NaN
2                               16.90
NaN
3                               23.10
NaN
4                               14.40
NaN

[5 rows x 33 columns]

print(f"Taille du DataFrame pivoté :
{df_pc_pivot_saison_median.shape[0]}")
```

Taille du DataFrame pivoté : 292196

Gestion des valeurs nulles

```
# afficher les valeurs nulles
df_pc_pivot_saison_median.isnull().sum()

param
station 0
année 0
saison 0
Ammonium - Air 290188
Ammonium - Eau 44641
Azote Kjeldahl - Air 290809
Azote Kjeldahl - Eau 42131
Carbone Organique - Eau 36462
Conductivité à 25°C - Air 289987
Conductivité à 25°C - Eau 6910
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Air 290115
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 30477
Diuron - Air 291423
Diuron - Eau 157613
Matières en suspension - Air 290114
Matières en suspension - Eau 12243
Nitrates - Air 290114
Nitrates - Eau 31344
Nitrites - Air 290114
Nitrites - Eau 35222
Orthophosphates (P04) - Air 290114
Orthophosphates (P04) - Eau 31970
Oxygène dissous - Air 289987
Oxygène dissous - Eau 12305
Phosphore total - Air 290187
Phosphore total - Eau 22900
Potentiel en Hydrogène (pH) - Air 289987
Potentiel en Hydrogène (pH) - Eau 5512
Taux de saturation en oxygène - Air 290154
Taux de saturation en oxygène - Eau 18128
Température de l'Eau - Air 289985
Température de l'Eau - Eau 5620
Turbidité Formazine Néphéломétrique - Eau 95299
dtype: int64

# pourcentage de valeurs nulles pour chaque paramètre
null_percentages = df_pc_pivot_saison_median.isnull().mean() * 100
null_percentages

param
station 0.000000
année 0.000000
saison 0.000000
Ammonium - Air 99.312790
Ammonium - Eau 15.277759
```

Azote Kjeldahl - Air	99.525319
Azote Kjeldahl - Eau	14.418746
Carbone Organique - Eau	12.478610
Conductivité à 25°C - Air	99.244001
Conductivité à 25°C - Eau	2.364851
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Air	99.287807
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	10.430328
Diuron - Air	99.735452
Diuron - Eau	53.940848
Matières en suspension - Air	99.287465
Matières en suspension - Eau	4.189996
Nitrates - Air	99.287465
Nitrates - Eau	10.727046
Nitrites - Air	99.287465
Nitrites - Eau	12.054238
Orthophosphates (P04) - Air	99.287465
Orthophosphates (P04) - Eau	10.941286
Oxygène dissous - Air	99.244001
Oxygène dissous - Eau	4.211214
Phosphore total - Air	99.312448
Phosphore total - Eau	7.837205
Potentiel en Hydrogène (pH) - Air	99.244001
Potentiel en Hydrogène (pH) - Eau	1.886405
Taux de saturation en oxygène - Air	99.301154
Taux de saturation en oxygène - Eau	6.204055
Température de l'Eau - Air	99.243316
Température de l'Eau - Eau	1.923367
Turbidité Formazine Néphéломétrique - Eau	32.614752

dtype: float64

```
# Représentation des pourcentages de valeurs nulles par paramètre
null_percentages = df_pc_pivot_saison_median.isnull().mean() * 100
null_percentages_df = null_percentages.reset_index()
null_percentages_df.columns = ['Paramètre', 'Pourcentage de valeurs nulles']
null_percentages_df = null_percentages_df.sort_values(by='Pourcentage de valeurs nulles')
fig = px.bar(null_percentages_df, x='Paramètre', y='Pourcentage de valeurs nulles', title='Pourcentage de valeurs nulles par paramètre', labels={'Pourcentage de valeurs nulles': 'Pourcentage (%)', 'Paramètre': 'Paramètre'}, text='Pourcentage de valeurs nulles')
fig.update_traces(texttemplate='%{text:.2f}%', textposition='outside')
fig.update_layout(xaxis_tickangle=45, width=1000, height=600, showlegend=False)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [{"alignmentgroup": "True", "hovertemplate": "Paramètre=%{x}<br>Pourcentage (%)=%{text}<extra></extra>", "legendgroup": "", "marker": {}}, {"x": "Azote Kjeldahl - Air", "y": 0.525319}, {"x": "Azote Kjeldahl - Eau", "y": 0.01418746}, {"x": "Carbone Organique - Eau", "y": 0.01247861}, {"x": "Conductivité à 25°C - Air", "y": 0.99244001}, {"x": "Conductivité à 25°C - Eau", "y": 0.002364851}, {"x": "Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Air", "y": 0.99287807}, {"x": "Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau", "y": 0.010430328}, {"x": "Diuron - Air", "y": 0.99735452}, {"x": "Diuron - Eau", "y": 0.053940848}, {"x": "Matières en suspension - Air", "y": 0.99287465}, {"x": "Matières en suspension - Eau", "y": 0.004189996}, {"x": "Nitrates - Air", "y": 0.99287465}, {"x": "Nitrates - Eau", "y": 0.010727046}, {"x": "Nitrites - Air", "y": 0.99287465}, {"x": "Nitrites - Eau", "y": 0.012054238}, {"x": "Orthophosphates (P04) - Air", "y": 0.99287465}, {"x": "Orthophosphates (P04) - Eau", "y": 0.010941286}, {"x": "Oxygène dissous - Air", "y": 0.99244001}, {"x": "Oxygène dissous - Eau", "y": 0.004211214}, {"x": "Phosphore total - Air", "y": 0.99312448}, {"x": "Phosphore total - Eau", "y": 0.007837205}, {"x": "Potentiel en Hydrogène (pH) - Air", "y": 0.99244001}, {"x": "Potentiel en Hydrogène (pH) - Eau", "y": 0.001886405}, {"x": "Taux de saturation en oxygène - Air", "y": 0.99301154}, {"x": "Taux de saturation en oxygène - Eau", "y": 0.006204055}, {"x": "Température de l'Eau - Air", "y": 0.99243316}, {"x": "Température de l'Eau - Eau", "y": 0.001923367}, {"x": "Turbidité Formazine Néphéломétrique - Eau", "y": 0.032614752}]}]
```

```
{"color": "#636efa", "pattern": {"shape": ""}}, "name": "", "offsetgroup": "", "orientation": "v", "showlegend": false, "text": [0, 0, 0, 1.8864050158113048, 1.9233665074128325, 2.3648509904310804, 4.1899 95756273186, 4.21121439034073, 6.2040548125231005, 7.837205163657271, 10.4 3032758833112, 10.727046229243385, 10.941285986118906, 12.05423756656491, 12.478610247915782, 14.418746320962642, 15.277758764664814, 32.6147517419 8141, 53.94084792399623, 99.24331613026872, 99.24400060233542, 99.24400060 233542, 99.24400060233542, 99.28746457857055, 99.28746457857055, 99.287464 57857055, 99.28746457857055, 99.2878068146039, 99.30115401990444, 99.31244 780900491, 99.31279004503826, 99.52531862174705, 99.7354515462224], "textposition": "outside", "texttemplate": "%{text:.2f}%", "type": "bar", "x": ["station", "année", "saison", "Potentiel en Hydrogène (pH) - Eau", "Température de l'Eau - Eau", "Conductivité à 25°C - Eau", "Matières en suspension - Eau", "Oxygène dissous - Eau", "Taux de saturation en oxygène - Eau", "Phosphore total - Eau", "Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau", "Nitrates - Eau", "Orthophosphates (P04) - Eau", "Nitrites - Eau", "Carbone Organique - Eau", "Azote Kjeldahl - Eau", "Ammonium - Eau", "Turbidité Formazine Néphélométrique - Eau", "Diuron - Eau", "Température de l'Eau - Air", "Oxygène dissous - Air", "Conductivité à 25°C - Air", "Potentiel en Hydrogène (pH) - Air", "Nitrites - Air", "Orthophosphates (P04) - Air", "Matières en suspension - Air", "Nitrates - Air", "Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Air", "Taux de saturation en oxygène - Air", "Phosphore total - Air", "Ammonium - Air", "Azote Kjeldahl - Air", "Diuron - Air"], "xaxis": "x", "y": [0, 0, 0, 1.8864050158113048, 1.9233665074128325, 2.3648509904310804, 4.1899 95756273186, 4.21121439034073, 6.2040548125231005, 7.837205163657271, 10.4 3032758833112, 10.727046229243385, 10.941285986118906, 12.05423756656491, 12.478610247915782, 14.418746320962642, 15.277758764664814, 32.6147517419 8141, 53.94084792399623, 99.24331613026872, 99.24400060233542, 99.24400060 233542, 99.24400060233542, 99.28746457857055, 99.28746457857055, 99.287464 57857055, 99.28746457857055, 99.2878068146039, 99.30115401990444, 99.31244 780900491, 99.31279004503826, 99.52531862174705, 99.7354515462224], "yaxis": "y"}, "layout": {"barmode": "relative", "height": 600, "legend": {"tracegroupgap": 0}, "showlegend": false, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}]}, "contour":
```

```

[{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "contour"}, {"contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}]}, {"heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmap"}]}, {"heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmapgl"}]}, {"histogram": [{"marker": {"pattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}]}, {"histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2d"}]}, {"histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2dcontour"}]}, {"mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}]}, {"parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}]}, {"pie": [{"automargin": true, "type": "pie"}]}, {"scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter"}]}, {"scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}]}, {"scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}]}, {"scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}]}, {"scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}]}, {"scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}]}

```

```

[{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolar"}}, {"scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolargl"}}, {"scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterternary"}}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FEBC52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, {"scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white"}}

```

```

    "lor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}  

    , "zaxis":  

    {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}, "title": {"text": "Pourcentage de valeurs nulles par paramètre"}, "width": 1000, "xaxis": {"anchor": "y", "domain": [0, 1], "tickangle": 45, "title": {"text": "Paramètre"}}, "yaxis": {"anchor": "x", "domain": [0, 1], "title": {"text": "Pourcentage (%)"}}}}  

# liste des paramètres avec plus de 90% de valeurs nulles  

params_with_most_nulls = null_percentages=null_percentages > 90].index.tolist()  

params_with_most_nulls  

['Ammonium - Air',  

 'Azote Kjeldahl - Air',  

 'Conductivité à 25°C - Air',  

 'Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Air',  

 'Diuron - Air',  

 'Matières en suspension - Air',  

 'Nitrates - Air',  

 'Nitrites - Air',  

 'Orthophosphates (P04) - Air',  

 'Oxygène dissous - Air',  

 'Phosphore total - Air',  

 'Potentiel en Hydrogène (pH) - Air',  

 'Taux de saturation en oxygène - Air',  

 "Température de l'Eau - Air"]

```

Nous avons décidé de supprimer les colonnes contenant plus de 90% de valeurs nulles, car un pourcentage aussi élevé de valeurs manquantes indique que ces paramètres sont rarement mesurés pour une station donnée à une saison donnée.

```

# supprimer les colonnes avec plus de 90% de valeurs nulles  

df_pc_pivot_saison_median.drop(columns=params_with_most_nulls,  

inplace=True)

```

```

# Affichage du nombre de lignes où il n'y a que la saison et l'année
# qui ne sont pas nuls
# Et donc où toutes les valeurs des paramètres sont nuls
exclude_columns = ['station', 'année', 'saison']
columns_to_check = [col for col in df_pc_pivot_saison_median.columns
if col not in exclude_columns]
rows_with_all_nulls =
df_pc_pivot_saison_median[df_pc_pivot_saison_median[columns_to_check].
isnull().all(axis=1)]
rows_with_all_nulls.head(3)

param    station   année   saison   Ammonium - Eau   Azote Kjeldahl - Eau \
35512  03017000  2005   Automne           NaN               NaN
35513  03017000  2005   Hiver              NaN               NaN
35656  03024392  2005   Hiver              NaN               NaN

param  Carbone Organique - Eau  Conductivité à 25°C - Eau \
35512                  NaN               NaN
35513                  NaN               NaN
35656                  NaN               NaN

param  Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \
35512                      NaN
35513                      NaN
35656                      NaN

param  Diuron - Eau  Matières en suspension - Eau   Nitrates - Eau \
35512                  NaN               NaN
35513                  NaN               NaN
35656                  NaN               NaN

param  Nitrites - Eau  Orthophosphates (P04) - Eau  Oxygène dissous - Eau \
35512                      NaN               NaN
NaN
35513                      NaN               NaN
NaN
35656                      NaN               NaN
NaN

param  Phosphore total - Eau  Potentiel en Hydrogène (pH) - Eau \
35512                  NaN               NaN
35513                  NaN               NaN
35656                  NaN               NaN

param  Taux de saturation en oxygène - Eau  Température de l'Eau - Eau

```

```

\35512           NaN           NaN
35513           NaN           NaN
35656           NaN           NaN

param Turbidité Formazine Néphélométrique - Eau
35512           NaN           NaN
35513           NaN           NaN
35656           NaN           NaN

rows_with_all_nulls.shape
(517, 19)

```

Il y a environ 500 lignes ne contenant aucune information, nous les supprimons.

```

# supprimer les lignes où il n'y a que param saison et année qui ne
# sont pas nuls
print("Nombre de lignes avant suppression :",
df_pc_pivot_saison_median.shape[0])
df_pc_pivot_saison_median =
df_pc_pivot_saison_median[~df_pc_pivot_saison_median[columns_to_check]
.isnull().all(axis=1)]
print("Nombre de lignes après suppression :",
df_pc_pivot_saison_median.shape[0])

Nombre de lignes avant suppression : 292196
Nombre de lignes après suppression : 291679

```

Analyse des données agrégées avec la moyenne

```

df_pc_pivot_saison_mean.head(5)

param   station   année    saison  Ammonium - Air  Ammonium - Eau \
0      05001800  2005     Automne        NaN           NaN
1      05001800  2005      Hiver        NaN           NaN
2      05001800  2005  Printemps        NaN           NaN
3      05001800  2005      Été         NaN           NaN
4      05001800  2007     Automne        NaN        0.025

param Azote Kjeldahl - Air  Azote Kjeldahl - Eau  Carbone Organique -
Eau \
0                   NaN           NaN
NaN
1                   NaN           NaN
NaN
2                   NaN           NaN
NaN

```

3	NaN	NaN		
NaN				
4	NaN	1.0		
2.75				
param	Conductivité à 25°C - Air	Conductivité à 25°C - Eau	...	\
0	NaN	603.000000	...	
1	NaN	825.000000	...	
2	NaN	768.333333	...	
3	NaN	628.666667	...	
4	NaN	776.666667	...	
param	Oxygène dissous - Eau	Phosphore total - Air	Phosphore total - Eau	\
0	7.500000		NaN	
0.05				
1	11.950000		NaN	
0.05				
2	10.766667		NaN	
0.05				
3	7.733333		NaN	
0.05				
4	9.633333		NaN	
0.05				
param	Potentiel en Hydrogène (pH) - Air	Potentiel en Hydrogène (pH) - Eau	\	
0		NaN		
7.500000				
1		NaN		
7.900000				
2		NaN		
7.966667				
3		NaN		
7.700000				
4		NaN		
8.000000				
param	Taux de saturation en oxygène - Air	\		
0		NaN		
1		NaN		
2		NaN		
3		NaN		
4		NaN		
param	Taux de saturation en oxygène - Eau	Température de l'Eau - Air	\	
0	82.000000		NaN	
1	103.500000		NaN	

2	104.666667	NaN
3	89.000000	NaN
4	92.000000	NaN
param Température de l'Eau - Eau Turbidité Formazine Néphéломétrique		
- Eau		
0	19.600000	
NaN		
1	8.950000	
NaN		
2	14.800000	
NaN		
3	22.500000	
NaN		
4	11.766667	
NaN		
[5 rows x 33 columns]		

Nous pouvons faire le même constat que pour les données agrégées par médiane, et nous procédons donc au même nettoyage pour les données agrégées par moyenne.

```
# supprimer les colonnes avec plus de 90% de valeurs nulles
df_pc_pivot_saison_mean.drop(columns=params_with_most_nulls,
inplace=True)

# supprimer les lignes où il n'y a que param saison et année qui ne
# sont pas nuls
print("Nombre de lignes avant suppression :",
df_pc_pivot_saison_mean.shape[0])
df_pc_pivot_saison_mean =
df_pc_pivot_saison_mean[~df_pc_pivot_saison_mean[columns_to_check].isn
ull().all(axis=1)]
print("Nombre de lignes après suppression :",
df_pc_pivot_saison_mean.shape[0])

Nombre de lignes avant suppression : 292196
Nombre de lignes après suppression : 291679
```

Analyse exploratoire des données hydrobiologiques

Démarche

Dans cette partie, nous allons analyser et traiter les données hydrobiologiques afin de les préparer pour les intégrer correctement avec les données physicochimiques.

Nous avons conservé seulement les informations utiles pour notre analyse, notamment l'indicateur biologique I2M2, les identifiants des stations pour permettre la liaison avec les autres tables, et les dates de prélèvement.

Ensuite, nous avons structuré les données en les agrégeant par station, saison et année. Nous avons créé deux ensembles de données : l'un avec les médianes des paramètres pour chaque station sur une saison et une année données, et l'autre avec les moyennes. Cette agrégation a permis de réduire la taille globale des données, tout en diminuant la variabilité et en équilibrant le nombre d'enregistrements entre les périodes.

Et enfin, pour tenir compte du délai entre les changements physicochimiques et leur impact sur les indices biologiques, nous avons introduit des décalages temporels (lags) de 1 mois, 3 mois, 6 mois et 1 an. Pour chaque décalage, nous avons généré de nouveaux ensembles de datasets.

Nettoyage des données

```
df_hydrobio.head(3)
```

```
    Unnamed: 0 CdStationMesureEauxSurface
LbStationMesureEauxSurface \
0          0             2000990 LE LERTZBACH À HEGENHEIM
1          1             2001000 L 'AUGRABEN À BARTENHEIM
2          2             2001000 L 'AUGRABEN À BARTENHEIM

    CdPointEauxSurf DateDebutOperationPrelBio CdSupport \
0           1.0            2010-07-20          13
1           2.0            2010-09-20          13
2           2.0            2011-08-03          13

    LbSupport DtProdResultatBiologique \
0 Macroinvertébrés aquatiques           NaN
1 Macroinvertébrés aquatiques           NaN
2 Macroinvertébrés aquatiques           NaN

    CdParametreResultatBiologique
LbLongParametre \
0                         7613 Indice Invertébrés Multimétrique
(I2M2)
1                         7613 Indice Invertébrés Multimétrique
(I2M2)
2                         7613 Indice Invertébrés Multimétrique
(I2M2)

    ... CdUniteMesure SymUniteMesure CdRqIndiceResultatBiologique \
0   ...           X                 X                  1
1   ...           X                 X                  1
2   ...           X                 X                  1
```

```

MnemoRqAna CdMethEval \
0 Résultat > seuil de quantification et < au seu...      NaN
1 Résultat > seuil de quantification et < au seu...      NaN
2 Résultat > seuil de quantification et < au seu...      NaN

RefOperationPrelBio CdProducteur NomProducteur \
0 1814647 18570301400018 AGENCE DE L'EAU RHIN MEUSE
1 1814648 18570301400018 AGENCE DE L'EAU RHIN MEUSE
2 2148304 18570301400018 AGENCE DE L'EAU RHIN MEUSE

CdAccredRsIndiceResultatBiologique
MnAccredRsIndiceResultatBiologique
0 0.0
Inconnu
1 0.0
Inconnu
2 0.0
Inconnu

[3 rows x 21 columns]

df_hydrobio.shape

(43535, 21)

```

Nous pouvons déjà voir qu'il y a beaucoup moins de données hydrobiologiques par rapport aux données physicochimiques (8917443 lignes dans le dataset physicochimiques).

```

# Valeurs manquantes
df_hydrobio.isnull().sum()

Unnamed: 0 0
CdStationMesureEauxSurface 0
LbStationMesureEauxSurface 0
CdPointEauxSurf 420
DateDebutOperationPrelBio 0
CdSupport 0
LbSupport 0
DtProdResultatBiologique 28706
CdParametreResultatBiologique 0
LbLongParametre 0
ResIndiceResultatBiologique 13
CdUniteMesure 0
SymUniteMesure 0
CdRqIndiceResultatBiologique 0
MnemoRqAna 0
CdMethEval 15162
RefOperationPrelBio 0
CdProducteur 0
NomProducteur 4

```

```

CdAccredRsIndiceResultatBiologique      192
MnAccredRsIndiceResultatBiologique      192
dtype: int64

# Nombre de valeurs uniques pour chaque colonne
df_hydrobio.nunique()

Unnamed: 0                      43535
CdStationMesureEauxSurface          9489
LbStationMesureEauxSurface          9389
CdPointEauxSurf                     26
DateDebutOperationPrelBio           2366
CdSupport                           1
LbSupport                           1
DtProdResultatBiologique            54
CdParametreResultatBiologique       1
LbLongParametre                     1
ResIndiceResultatBiologique         9111
CdUniteMesure                        3
SymUniteMesure                       3
CdRqIndiceResultatBiologique        2
MnemoRqAna                           2
CdMethEval                           7
RefOperationPrelBio                 43535
CdProducteur                         262
NomProducteur                        248
CdAccredRsIndiceResultatBiologique   3
MnAccredRsIndiceResultatBiologique   3
dtype: int64

# Recherche des correspondances multiples entre deux colonnes
def afficher_correspondances_multiples(df, col_code, col_libelle):
    multiples = df.groupby(col_code)[col_libelle].nunique()
    codes_avec_multiples_libelles = multiples[multiples > 1].index

    if len(codes_avec_multiples_libelles) > 0:
        print(f"Codes dans '{col_code}' avec plusieurs valeurs dans '{col_libelle}' :")
        print(df[df[col_code].isin(codes_avec_multiples_libelles)][[col_code, col_libelle]].drop_duplicates())
    else:
        print(f"Aucune correspondance multiple trouvée entre '{col_code}' et '{col_libelle}'.")

# Vérification des unités de mesure
print("Valeurs uniques CdUniteMesure :",
df_hydrobio['CdUniteMesure'].unique())
print("Valeurs uniques SymUniteMesure :",
df_hydrobio['SymUniteMesure'].unique())

```

```

afficher_correspondances_multiples(df_hydrobio, 'CdUniteMesure',
'SymUniteMesure')

Valeurs uniques CdUniteMesure :  ['X' '214' '0']
Valeurs uniques SymUniteMesure :  ['X' 'n' 'Unité inconnue']
Aucune correspondance multiple trouvée entre 'CdUniteMesure' et
'SymUniteMesure'.

afficher_correspondances_multiples(df_hydrobio,
'CdStationMesureEauxSurface', 'LbStationMesureEauxSurface')

Codes dans 'CdStationMesureEauxSurface' avec plusieurs valeurs dans
'LbStationMesureEauxSurface' :
      CdStationMesureEauxSurface    LbStationMesureEauxSurface
2588            6073500    LEYSSE A LE-BOURGET-DU-LAC
2986            6135500        ARLY A FLUMET
4066            6580830      DEISSE A GRESY-SUR-AIX
42528           6073500    LEYSSE A LE-BOURGET-DU-LAC 1
42750           6135500        ARLY A FLUMET 1
43397           6580830      DEISSE A GRESY-SUR-AIX 1

# Renommage des stations ayant le même CdStationMesureEauxSurface et
des libellés un peu différents
codes_a_corriger = [6073500, 6135500, 6580830]
df_hydrobio.loc[df_hydrobio['CdStationMesureEauxSurface'].isin(codes_a_
_corriger), 'LbStationMesureEauxSurface'] = \
df_hydrobio.loc[df_hydrobio['CdStationMesureEauxSurface'].isin(codes_a_
_corriger), 'LbStationMesureEauxSurface'].str.rstrip(' 1')

# Vérification
afficher_correspondances_multiples(df_hydrobio,
'CdStationMesureEauxSurface', 'LbStationMesureEauxSurface')

Aucune correspondance multiple trouvée entre
'CdStationMesureEauxSurface' et 'LbStationMesureEauxSurface'.

```

Une grande partie des colonnes de ce jeu de données ne sont pas pertinentes pour l'analyse et nous les supprimerons par la suite.

Voici les colonnes identifiées comme inutiles :

- **Unnamed: 0** : Colonne d'index sans valeur analytique.
- **DtProdResultatBiologique** : Date (jour, mois, année) de calcul du résultat biologique, non utile pour notre analyse et contenant des valeurs manquantes dans 1/4 des cas.
- **CdSupport, LbSupport, CdParametreResultatBiologique, LbLongParametre** : Colonnes avec une seule valeur possible et aucune donnée manquante, donc non informatives.
- **RefOperationPrelBio** : Sert uniquement à des jointures avec des tables externes que nous n'utilisons pas.

- CdAccredRsIndiceResultatBiologique, CdProducteur, NomProducteur, MnAccredRsIndiceResultatBiologique : Métadonnées administratives sans utilité pour notre analyse.
- CdUniteMesure, SymUniteMesure : Colonnes contenant trois valeurs possibles, mais toutes indiquant l'absence d'unité, donc sans intérêt analytique.

```
df_hydrobio.rename(columns={'CdStationMesureEauxSurface': 'station',
                           'DateDebutOperationPrelBio': 'date', 'ResIndiceResultatBiologique':
                           'I2M2'}, inplace=True)

df_hydrobio['date'] = pd.to_datetime(df_hydrobio['date'], format='%Y-%m-%d')
df_hydrobio['date'].dtype

dtype('<M8[ns]')

# supprimer les I2M2 manquants et les doublons
print("avant suppression :", df_hydrobio.shape)
df_hydrobio.dropna(subset=['I2M2'], inplace=True)
df_hydrobio.drop_duplicates(inplace=True)
print("après suppression :", df_hydrobio.shape)

avant suppression : (43535, 21)
après suppression : (43522, 21)

# Gestion des seuils
df_hydrobio['MnemoRqAna'].value_counts()

MnemoRqAna
Résultat > seuil de quantification et < au seuil de saturation
43522
Name: count, dtype: int64
```

Le seuil est respecté pour l'ensemble des données.

```
# Statistiques descriptives de I2M2
desc_stats = df_hydrobio['I2M2'].describe()
desc_stats.drop(['count'], inplace=True)
fig = go.Figure()
fig.add_trace(go.Bar(x=desc_stats.index, y=desc_stats.values))
fig.update_layout(height=500, width=400, font_size=10,
                  title="Statistique descriptives I2M2", showlegend=False)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
    {"type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.5229396530848766, 0.23793162147026586, 0, 0.347025, 0.558, 0.715, 1]}, {"layout": {"font": {"size": 10}, "height": 500, "showlegend": false, "template": [{"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}}, "pattern": null}}]}]}
```

```

  {"fillmode": "overlay", "size": 10, "solidity": 0.2}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "contour"}], "contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmapgl"}], "histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}], "histogram2d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2d"}], "histogram2dcontour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"automargin": true, "type": "pie"}], "scatter": [{"fillpattern": "

```

```

{"fillmode": "overlay", "size": 10, "solidity": 0.2}, {"type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlinewidth": 0, "ticks": ""}}, {"type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}, "type": "table"}}, "layout": {"annotationdefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbcb4"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}, {"colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECKB52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"center": {"lat": 40, "lon": -95}, "zoom": 3}, "projection": "mercator", "style": "light"}, {"type": "surface"}]}], "type": "surface"}]

```

```
{
  "style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "plot": {
    "angularaxis": {
      "gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis": {
      "gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {
    "xaxis": {
      "backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "yaxis": {
      "backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "zaxis": {
      "backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "bgcolor": "#E5ECF6", "width": 400}
  },
  "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}}, "title": {"text": "Statistique descriptives I2M2"}, "width": 400}
}
```

Le paramètre I2M2 ne comporte pas d'outliers, toutes les valeurs sont comprises entre 0 et 1.

Aggrégation par saison

```
df_hydrobio_saison = df_hydrobio.copy()
df_hydrobio_saison['année'] = df_hydrobio_saison['date'].dt.year +
(df_hydrobio_saison['date'].dt.month == 12)
df_hydrobio_saison['saison'] =
df_hydrobio_saison['date'].dt.month.map({
    12: 'Hiver', 1: 'Hiver', 2: 'Hiver',
    3: 'Printemps', 4: 'Printemps', 5: 'Printemps',
    6: 'Été', 7: 'Été', 8: 'Été',
    9: 'Automne', 10: 'Automne', 11: 'Automne'
})
df_hydrobio_saison = df_hydrobio_saison[['station', 'année', 'saison',
'I2M2']]
df_hydrobio_saison
```

	station	année	saison	I2M2
0	2000990	2010	Été	0.4726
1	2001000	2010	Automne	0.3481
2	2001000	2011	Été	0.4253

```

3      2001000    2012     Été  0.2460
4      2001025    2010     Été  0.1137
...
43530  6999107    2009     Été  0.1820
43531  6999125    2008     Été  0.1180
43532  6999125    2009     Été  0.1580
43533  6999180    2008     Été  0.3530
43534  6999180    2009     Été  0.4150

[43522 rows x 4 columns]

df_counts = df_hydrobio_saison.groupby(['année',
'saison']).size().reset_index(name='nombre de résultats d\'analyse par saison')

```

Observation des données

```

fig = px.bar(df_counts, x='année', y="nombre de résultats d'analyse par saison", color='saison', barmode='stack', labels={'Année': 'année', 'Nombre de valeurs': 'Nombre de Mesures'}, title='Nombre de mesures par année et saison', color_discrete_map=season_colors)
fig.update_layout(xaxis_title='année', yaxis_title="nombre de résultats d'analyse", legend_title='saison')
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [{"alignmentgroup": "True", "hovertemplate": "saison=Automne<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra></extra>", "legendgroup": "Automne", "marker": {"color": "#FF6961", "pattern": {"shape": ""}}, "name": "Automne", "offsetgroup": "Automne", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [61, 223, 157, 220, 247, 331, 487, 586, 607, 864, 807, 826, 617, 598, 746, 386], "yaxis": "y"}, {"alignmentgroup": "True", "hovertemplate": "saison=Été<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra></extra>", "legendgroup": "Été", "marker": {"color": "#FFB347", "pattern": {"shape": ""}}, "name": "Été", "offsetgroup": "Été", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [102, 772, 906, 997, 1038, 1099, 1755, 1471, 3125, 2981, 2712, 3084, 2857, 3228, 2969, 2612], "yaxis": "y"}, {"alignmentgroup": "True", "hovertemplate": "saison=Hiver<br>année=%{x}<br>nombre de résultats d'analyse par saison=%{y}</extra></extra>", "legendgroup": "Hiver", "marker": {"color": "#AEC6CF", "pattern": {"shape": ""}}, "name": "Hiver", "offsetgroup": "Hiver", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022], "xaxis": "x", "y": [102, 772, 906, 997, 1038, 1099, 1755, 1471, 3125, 2981, 2712, 3084, 2857, 3228, 2969, 2612], "yaxis": "y"}]

```

```

showlegend":true,"textposition":"auto","type":"bar","x":
[2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021
,2022,2023],"xaxis":"x","y":
[30,54,46,54,63,39,42,91,32,42,68,84,57,74,81,3],"yaxis":"y"},
{"alignmentgroup":"True","hovertemplate":"saison=Printemps<br>année=%
{x}<br>nombre de résultats d'analyse par
saison=%{y}<extra></extra>","legendgroup":"Printemps","marker":
{"color":"#77DD77","pattern":
{"shape":""}}, "name":"Printemps","offsetgroup":"Printemps","orientatio
n":"v","showlegend":true,"textposition":"auto","type":"bar","x":
[2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021
,2022],"xaxis":"x","y":
[34,49,52,105,41,60,122,192,124,276,232,424,269,577,634],"yaxis":"y"}]
,"layout": {"barmode": "stack", "legend": {"title": {"text": "saison"}, "tracegroupgap": 0}, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"]}, "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"]}, {"carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min
orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min
orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, {"choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}]}, {"contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "contour"}], "contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet"}]}, {"heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmapgl"}], "histogram": [{"marker": {"pattern": "

```

```

  {"fillmode": "overlay", "size": 10, "solidity": 0.2}], "type": "histogram"}],  

  "histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale":  

    [[0, "#0d0887"], [0.1111111111111111, "#46039f"],  

     [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],  

     [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],  

     [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],  

     [0.8888888888888888, "#fdca26"],  

     [1, "#f0f921"]]}, {"type": "histogram2d"}], "histogram2dcontour":  

  [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale":  

    [[0, "#0d0887"], [0.1111111111111111, "#46039f"],  

     [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],  

     [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],  

     [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],  

     [0.8888888888888888, "#fdca26"],  

     [1, "#f0f921"]]}, {"type": "histogram2dcontour"}], "mesh3d": [{"colorbar":  

    {"outlineWidth": 0, "ticks": ""}, {"type": "mesh3d"}], "parcoords": [{"line":  

    {"colorbar": {"outlineWidth": 0, "ticks": ""}}, {"type": "parcoords"}], "pie":  

  [{"automargin": true, "type": "pie"}], "scatter": [{"fillpattern":  

    {"fillmode": "overlay", "size": 10, "solidity": 0.2}, {"type": "scatter"}], "scatter3d":  

  [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker":  

    {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary":  

  [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface":  

  [{"colorbar": {"outlineWidth": 0, "ticks": ""}}, {"type": "surface"}], "table": [{"cells": {"fill":  

      {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill":  

        {"color": "#C8D4E3"}, "line":  

        {"color": "white"}}, "type": "table"}}], "layout": {"annotationdefaults":  

  {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers":  

  "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging":
```

```

[[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"],  

[0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"],  

[0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential":  

[[[0, "#0d0887"], [0.1111111111111111, "#46039f"],  

[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],  

[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],  

[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],  

[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus":  

[[[0, "#0d0887"], [0.1111111111111111, "#46039f"],  

[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],  

[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],  

[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],  

[0.8888888888888888, "#fdca26"], [1, "#f0f921"]]], "colorway":  

[{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692",  

"#B6E880", "#FF97FF", "#FECB52"}, "font": {"color": "#2a3f5f"}, "geo":  

{"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true,  

"showland": true, "subunitcolor": "white"}, "hoverlabel":  

{"align": "left"}, "hovermode": "closest", "mapbox":  

{"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar":  

{"angularaxis":  

{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis":  

{"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene":  

{"xaxis":  

{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  

"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis":  

{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  

"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis":  

{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  

"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "shapedefaults":  

{"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis":  

{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis":  

{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "caxis":  

{"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title":  

{"x": 5.0e-2}, "xaxis":  

{"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  

{"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis":  

{"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  

{"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}], "title":  

{"text": "Nombre de mesures par année et saison"}, "xaxis":  

{"anchor": "y", "domain": [0, 1], "title": {"text": "année"}}, "yaxis":  

{"anchor": "x", "domain": [0, 1], "title": {"text": "nombre de résultats  
d'analyse"}}]}

```

Nous pouvons voir que la répartition des prélèvements est très différente que celle des données physicochimiques qui étaient uniformément réparties sur toutes les saisons.

Ici, nous constatons une forte dominance des prélèvements réalisés en été. Les autres saisons sont beaucoup moins représentées, avec très peu de résultats enregistrés en hiver, un nombre légèrement supérieur au printemps, et encore légèrement supérieur en automne.

Avant 2008, le nombre de prélèvements était inférieur à 1 000 par an. Jusqu'en 2014, ce nombre a progressivement augmenté à moins de 2 000 par an, pour finalement doubler après 2014, avec une concentration importante des prélèvements en été.

```
# Seuils de l'I2M2
seuils_I2M2 = {
    "Très bon": 0.665,
    "Bon": 0.443,
    "Moyen": 0.295,
    "Médiocre": 0.148,
    "Mauvais": 0.0
}
# trouvés ici : https://www.laboceia.fr/indice-invertebres-multi-metriques-i2m2/?cn-reloaded=1

# Statistiques descriptives
desc_stats_saison = df_hydrobio_saison.groupby('saison')
['I2M2'].describe()
desc_stats_saison =
desc_stats_saison.drop(columns=['count']).transpose()
fig = go.Figure()
for saison in desc_stats_saison.columns:
    fig.add_trace(go.Bar(x=desc_stats_saison.index,
y=desc_stats_saison[saison],name=saison))
for etat, valeur in seuils_I2M2.items():
    fig.add_shape(type="line", x0=-0.5, x1=len(desc_stats)-0.5,y0=valeur, y1=valeur, line=dict(color="black", width=1,
dash="dash"), name=etat)
    fig.add_annotation(x=len(desc_stats)-0.5, y=valeur, text=etat,
showarrow=False, font=dict(size=10, color="black"), xanchor="left" )
fig.update_layout(height=500, width=700, font_size=10,
title="Statistiques descriptives de I2M2 par Saison",
xaxis_title="Statistique", yaxis_title="Valeur",barmode='group' )
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
{"name": "Automne", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.6005555929099575, 0.23132913543158254, 0, 0.4542, 0.6552, 0.78225, 1]}, {"name": "Hiver", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.599566511627907, 0.1648957003950427, 1.95e-2, 0.497, 0.6225, 0.723, 0.955]}, {"name": "Printemps", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.599566511627907, 0.1648957003950427, 1.95e-2, 0.497, 0.6225, 0.723, 0.955]} ]}
```

```

[0.4306876214384206, 0.2297334925347392, 0, 0.2484, 0.4361, 0.613, 0.994]}, {"name": "Été", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.5111427751289895, 0.236491464804705, 0, 0.335, 0.544, 0.703, 1]}], "layout": [{"annotations": [{"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Très bon", "x": 6.5, "xanchor": "left", "y": 0.665}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Bon", "x": 6.5, "xanchor": "left", "y": 0.443}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Moyen", "x": 6.5, "xanchor": "left", "y": 0.295}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Médiocre", "x": 6.5, "xanchor": "left", "y": 0.148}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Mauvais", "x": 6.5, "xanchor": "left", "y": 0}], "barmode": "group", "font": {"size": 10}, "height": 500, "shapes": [{"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Très bon", "type": "line", "x0": -0.5, "x1": 6.5, "y0": 0.665, "y1": 0.665}, {"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Bon", "type": "line", "x0": -0.5, "x1": 6.5, "y0": 0.443, "y1": 0.443}, {"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Moyen", "type": "line", "x0": -0.5, "x1": 6.5, "y0": 0.295, "y1": 0.295}, {"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Médiocre", "type": "line", "x0": -0.5, "x1": 6.5, "y0": 0.148, "y1": 0.148}, {"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Mauvais", "type": "line", "x0": -0.5, "x1": 6.5, "y0": 0, "y1": 0}], "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min": 0, "orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min": 0, "orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contour"}], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "heatmap"}]}], "colors": [{"color": "#0d0887", "value": 0}, {"color": "#46039f", "value": 0.1111111111111111}, {"color": "#7201a8", "value": 0.2222222222222222}, {"color": "#9c179e", "value": 0.3333333333333333}, {"color": "#bd3786", "value": 0.4444444444444444}, {"color": "#d8576b", "value": 0.5555555555555556}, {"color": "#ed7953", "value": 0.6666666666666666}, {"color": "#fb9f3a", "value": 0.7777777777777778}, {"color": "#fdca26", "value": 0.8888888888888888}, {"color": "#f0f921", "value": 1}], "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "heatmap"}]}], "heatmaps": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "heatmap"}]}], "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"]], "contour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contour"}], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "heatmap"}]}]

```

```

[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],
[1, "#f0f921]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "heatmapgl"}], "histogram": [{"marker": {"pattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}], "histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "histogram2d"}], "histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"autoMargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}}, {"type": "surface"}], [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"]}], [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"]]}]

```

```

[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],
[1, "#f0f921"]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}]}, "layout": {"annotationdefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbcc4"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]]}, "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FEBC52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, {"shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, {"title": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}], "x": 5.0e-2}, "xaxis": 

```

```

{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15,"zerolinecolor":"white","zerolinewidth":2}, "yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15,"zerolinecolor":"white","zerolinewidth":2}}}, "title":
{"text": "Statistiques descriptives de I2M2 par
Saison"}, "width":700, "xaxis": {"title": {"text": "Statistique"}}, "yaxis":
{"title": {"text": "Valeur"}}}}

```

Nous pouvons voir que les saisons influencent clairement l'indice biologique I2M2, avec une qualité biologique généralement plus favorable en été et en automne. Ce résultat pourrait être lié à des variations naturelles des conditions environnementales, des paramètres physicochimiques, ou à des variations des activités humaines impactant la qualité de l'eau.

Ajout d'un décalage temporelle

Pour notre analyse, nous avons introduit un décalage temporel initial arbitraire de 1 mois entre les données physicochimiques et hydrobiologiques.

Ce choix repose sur l'hypothèse que les variations des paramètres physicochimiques influencent l'indice I2M2, mais que cet impact n'est pas immédiat. En effet, les changements physicochimiques, comme une hausse de la température ou une variation des nutriments, nécessitent un certain temps pour se refléter dans la composition biologique de l'eau.

Nous prévoyons également de tester des décalages supplémentaires de 3 mois, 6 mois et 1 an, afin d'explorer l'effet différé des variations physicochimiques sur la qualité biologique, et d'identifier le délai optimal reflétant le mieux ces interactions.

```

# Puisque les données physicochimiques sont censées avoir un impact
sur les données biologiques,
# on simule un "retour dans le temps" pour les données
hydrobiologiques
# c'est à dire que si on fait un relevé en décembre des données
hydrobiologiques,
# on veut l'associer aux données de novembre des données
physicochimiques
# si le lag est de 1 mois.
df_hydrobio['date_lag_1_month'] = df_hydrobio['date'] -
pd.Timedelta(days=30)
df_hydrobio['date_lag_3_month'] = df_hydrobio['date'] -
pd.Timedelta(days=90)
df_hydrobio['date_lag_6_month'] = df_hydrobio['date'] -
pd.Timedelta(days=180)

df_hydrobio_lag_1_month = df_hydrobio.copy()
df_hydrobio_lag_3_month = df_hydrobio.copy()
df_hydrobio_lag_6_month = df_hydrobio.copy()

```

```

df_hydrobio_lag_1_month['année'] =
df_hydrobio_lag_1_month['date_lag_1_month'].dt.year +
(df_hydrobio_lag_1_month['date_lag_1_month'].dt.month == 12)
df_hydrobio_lag_1_month['saison'] =
df_hydrobio_lag_1_month['date_lag_1_month'].dt.month.map({
    12: 'Hiver', 1: 'Hiver', 2: 'Hiver',
    3: 'Printemps', 4: 'Printemps', 5: 'Printemps',
    6: 'Été', 7: 'Été', 8: 'Été',
    9: 'Automne', 10: 'Automne', 11: 'Automne'
})

df_hydrobio_lag_3_month['année'] =
df_hydrobio_lag_3_month['date_lag_3_month'].dt.year +
(df_hydrobio_lag_3_month['date_lag_3_month'].dt.month == 12)
df_hydrobio_lag_3_month['saison'] =
df_hydrobio_lag_3_month['date_lag_3_month'].dt.month.map({
    12: 'Hiver', 1: 'Hiver', 2: 'Hiver',
    3: 'Printemps', 4: 'Printemps', 5: 'Printemps',
    6: 'Été', 7: 'Été', 8: 'Été',
    9: 'Automne', 10: 'Automne', 11: 'Automne'
})

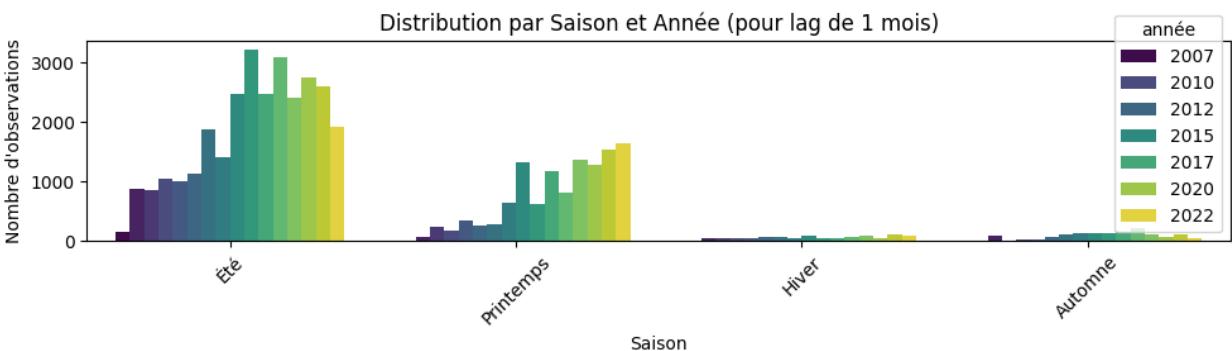
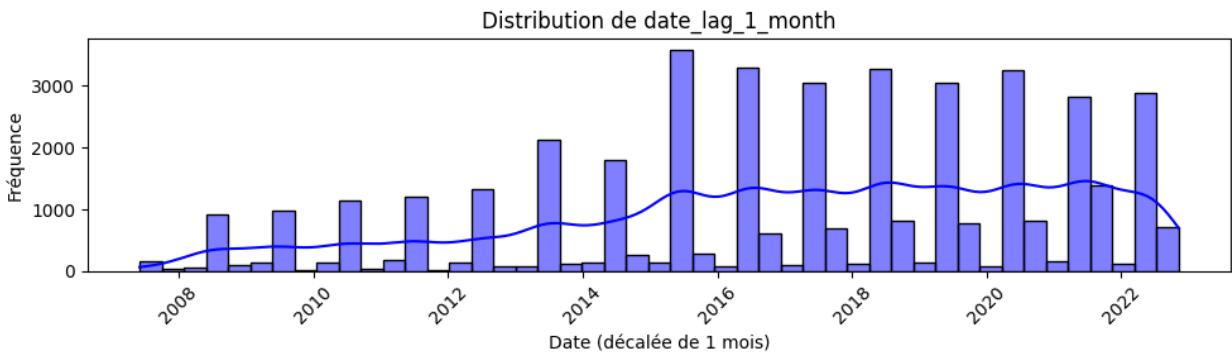
df_hydrobio_lag_6_month['année'] =
df_hydrobio_lag_6_month['date_lag_6_month'].dt.year +
(df_hydrobio_lag_6_month['date_lag_6_month'].dt.month == 12)
df_hydrobio_lag_6_month['saison'] =
df_hydrobio_lag_6_month['date_lag_6_month'].dt.month.map({
    12: 'Hiver', 1: 'Hiver', 2: 'Hiver',
    3: 'Printemps', 4: 'Printemps', 5: 'Printemps',
    6: 'Été', 7: 'Été', 8: 'Été',
    9: 'Automne', 10: 'Automne', 11: 'Automne'
})

# Distribution de df_hydrobio['date_lag_1_month']
plt.figure(figsize=(10, 3))
sns.histplot(df_hydrobio['date_lag_1_month'], kde=True, color='blue')
plt.title('Distribution de date_lag_1_month')
plt.xlabel('Date (décalée de 1 mois)')
plt.ylabel('Fréquence')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

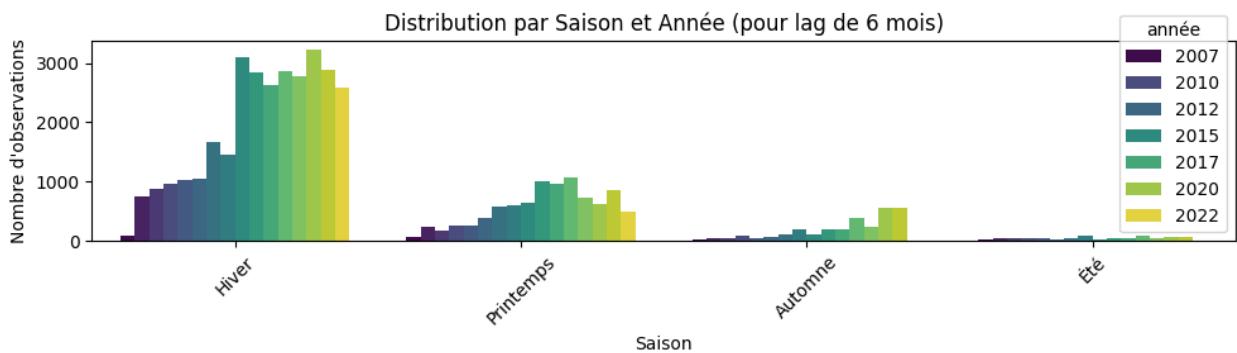
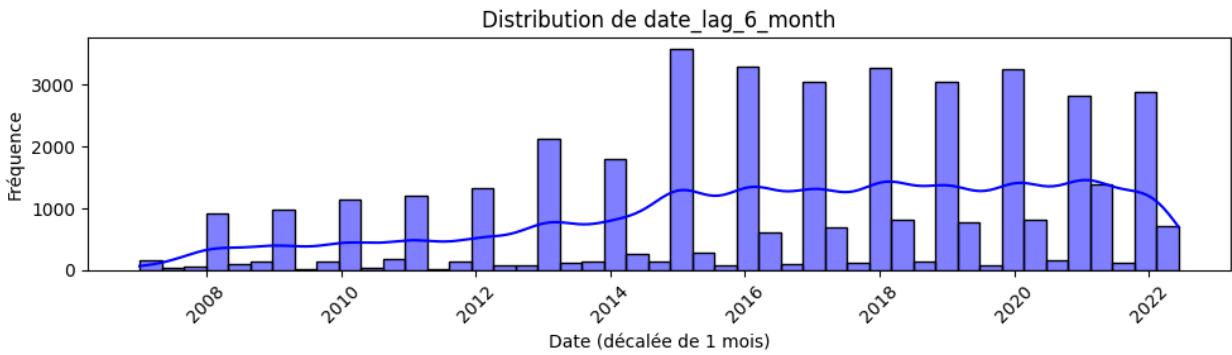
# Distribution par saison et année
plt.figure(figsize=(10, 3))
sns.countplot(x='saison', hue='année', data=df_hydrobio_lag_1_month,
palette='viridis')
plt.title('Distribution par Saison et Année (pour lag de 1 mois)')
plt.xlabel('Saison')
plt.ylabel('Nombre d\'observations')
plt.xticks(rotation=45)

```

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



```
# Distribution de df_hydrobio['date_lag_1_month']
plt.figure(figsize=(10, 3))
sns.histplot(df_hydrobio['date_lag_6_month'], kde=True, color='blue')
plt.title('Distribution de date_lag_6_month')
plt.xlabel('Date (décalée de 1 mois)')
plt.ylabel('Fréquence')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
# Distribution par saison et année
plt.figure(figsize=(10, 3))
sns.countplot(x='saison', hue='année', data=df_hydrobio_lag_6_month,
palette='viridis')
plt.title('Distribution par Saison et Année (pour lag de 6 mois)')
plt.xlabel('Saison')
plt.ylabel('Nombre d\'observations')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Les lags ont bien l'air effectifs.

```

df_hydrobio_lag_1_month_median =
df_hydrobio_lag_1_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'median'}).reset_index()
df_hydrobio_lag_3_month_median =
df_hydrobio_lag_3_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'median'}).reset_index()
df_hydrobio_lag_6_month_median =
df_hydrobio_lag_6_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'median'}).reset_index()
df_hydrobio_lag_1_month_mean =
df_hydrobio_lag_1_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'mean'}).reset_index()
df_hydrobio_lag_3_month_mean =
df_hydrobio_lag_3_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'mean'}).reset_index()
df_hydrobio_lag_6_month_mean =
df_hydrobio_lag_6_month.groupby(['station', 'année',
'saison']).agg({'I2M2': 'mean'}).reset_index()

df_hydrobio_lag_1_month_median.head(5)

```

	station	année	saison	I2M2
0	1000274	2016	Été	0.243
1	1000274	2017	Printemps	0.231
2	1000274	2018	Été	0.205

3	1000274	2019	Printemps	0.236
4	1000274	2020	Été	0.096

Intégration des données : Jointure des tables physicochimiques, hydrobiologiques, stations

Démarche

Nous avons joint les différentes tables de données (physicochimiques, hydrobiologiques, stations et hydroécorégions) pour constituer un dataset complet destiné aux analyses.

La première étape a consisté à nettoyer les colonnes relatives aux stations (`CdStationMesureEauxSurface` et `LbStationMesureEauxSurface`) dans chaque dataset, en corrigeant les identifiants et en vérifiant la cohérence des libellés. Ce nettoyage a été appliqué à l'ensemble des tables pour garantir leur compatibilité.

Nous avons ensuite utilisé le dataset `df_pc_median_bio_median_1_month`, qui regroupe les données physicochimiques et hydrobiologiques agrégées par médiane, avec un décalage temporel de 1 mois pour les indices hydrobiologiques, comme base pour toutes les étapes suivantes de l'analyse.

Enfin, le dataset nettoyé `df_pc_median_bio_median_1_month` a été joint aux données des hydroécorégions en fonction des coordonnées géographiques des stations, permettant ainsi de localiser les stations dans leurs hydroécorégions respectives.

Observation et nettoyage des colonnes relatives aux stations

```
# copie des dataframes pour nettoyer les colonnes station
# on garde juste cd et lb pour les stations
df_stations_join = df_stations.copy()
df_stations_join = df_stations_join[['CdStationMesureEauxSurface',
'LbStationMesureEauxSurface']].copy()
df_pc_join = df_pc.copy()
df_pc_join = df_pc[['CdStationMesureEauxSurface',
'LbStationMesureEauxSurface']].copy()
df_hydrobio_join = df_hydrobio.copy()
df_hydrobio_join = df_hydrobio[['station',
'LbStationMesureEauxSurface']].copy()
df_hydrobio_join.rename(columns={'station':
'CdStationMesureEauxSurface'}, inplace=True)

# afficher un échantillon des station pour chaque dataset
print("Stations dans df_stations")
print(df_stations_join['CdStationMesureEauxSurface'].dtype)
print(df_stations_join['CdStationMesureEauxSurface'].unique())
print()
print("Stations dans df_pc")
print(df_pc_join['CdStationMesureEauxSurface'].dtype)
print(df_pc_join['CdStationMesureEauxSurface'].unique())
```

```

print()
print("Stations dans df_hydrobio")
print(df_hydrobio_join['CdStationMesureEauxSurface'].dtype)
print(df_hydrobio_join['CdStationMesureEauxSurface'].unique())

Stations dans df_stations
object
['01000477' '01000602' '01000605' ... 'Y9205023' 'Y9715083'
 'Y9905043']

Stations dans df_pc
category
['05005600', '05200115', '05001800', '05004000', '05005000', ...,
 '06001219', '03137810', '03271785', '03113000', '06148115']
Length: 8809
Categories (8810, object): ['05001800', '05004000', '05005000',
 '05005350', ..., '06001219', '03271785', '03113000', '06148115']

Stations dans df_hydrobio
int64
[2000990 2001000 2001025 ... 6580040 6710037 6820126]

```

Ces colonnes doivent être modifiées pour être compatibles.

Selon la documentation, les codes des stations devraient être des entiers composés de 8 chiffres.

Nous avons donc décidé de nettoyer les identifiants pour respecter ce format.

```

def clean_station_column(df, column_name):
    df[column_name] = df[column_name].astype(str).str.extract(r'(\d+')[0]
    df[column_name] = pd.to_numeric(df[column_name], errors='coerce')
    return df
df_stations_join = clean_station_column(df_stations_join,
                                         'CdStationMesureEauxSurface')
df_pc_join = clean_station_column(df_pc_join,
                                   'CdStationMesureEauxSurface')
df_hydrobio_join = clean_station_column(df_hydrobio_join,
                                         'CdStationMesureEauxSurface')

# on crée un dataframe avec l'identifiant de chaque station, et pour
# les 3 datasets
# on affiche le libellé associé, de cette manière on peut vérifier si
# les stations sont les mêmes
merged_data = pd.merge(df_pc_join, df_hydrobio_join,
                       on='CdStationMesureEauxSurface', how='inner', suffixes=('_pc',
 '_hydrobio'))
merged_all = pd.merge(merged_data, df_stations_join,
                      on='CdStationMesureEauxSurface', how='inner')
# caster tous les Lb en object

```

```

merged_all['LbStationMesureEauxSurface_pc'] =
merged_all['LbStationMesureEauxSurface_pc'].astype(str).str.strip()
merged_all['LbStationMesureEauxSurface'] =
merged_all['LbStationMesureEauxSurface'].astype(str).str.strip()
merged_all['LbStationMesureEauxSurface_hydrobio'] =
merged_all['LbStationMesureEauxSurface_hydrobio'].astype(str).str.strip()
merged_all.dtypes

CdStationMesureEauxSurface           int64
LbStationMesureEauxSurface_pc        object
LbStationMesureEauxSurface_hydrobio  object
LbStationMesureEauxSurface          object
dtype: object

```

Étude des cas où les libellés sont différents

```

label_diff_pc_hydrobio =
merged_all[merged_all['LbStationMesureEauxSurface_pc'] != merged_all['LbStationMesureEauxSurface_hydrobio']]
label_diff_pc_hydrobio = label_diff_pc_hydrobio.drop_duplicates()
label_diff_pc_hydrobio

CdStationMesureEauxSurface \
142208      5197200
468800      6083000
490122      6800003
645026      6208900
1980639     6580640
2024300     6139405
2595740     5068920
6148279     6830030
10095956    6080995
13028690    6014250
15703808    6213230
16150583    3250952
26405099    6009020

LbStationMesureEauxSurface_pc \
142208      Le Majesq à Azur
468800      BOURBRE A CHAVANOZ
490122      EYGUES A ST-MAURICE/EYGUES - LES CIVARDIERES
645026      MOURACHONNE A PEGOMAS
1980639     BIEF D'ENFER A ST ETIENNE SUR REYSSOUZE
2024300     BUGEON A LA-CHAMBRE
2595740     La Véronne en aval de Riom-ès-Montagnes (Amont...
6148279     TORRENSON A ST-CYR
10095956    AGNY A NIVOLAS-VERMELLE
13028690    SANSFOND A SAULON-LA-RUE
15703808    RUISSEAU LE THOUX A CURIS AU MONT D'OR

```

16150583 LE COURS D'EAU NUMÉRO 01 DE LA BÉLINIÈRE A CON...
 26405099 REAL MARTIN A PIGNANS

	LbStationMesureEauxSurface_hydrobio \
142208	Le ruisseau de Magescq à Azur
468800	BOURBRE A CHAVANOZ 1
490122	EYGUES A ST-MAURICE-SUR-EYGUES 1
645026	MOURACHONNE A PEGOMAS 1
1980639	BIEF D'ENFER A ST ETIENNE SUR REYSSOUZE 1
2024300	BUGEON A LA-CHAMBRE 1
2595740	La Véronne en aval de Riom-ès-Montagnes (Amo...)
6148279	TORRENSON A THORRENC
10095956	AGNY A NIVOLAS-VERMELLE 1
13028690	CENT FONDS A SAULON-LA-RUE 1
15703808	RUISSEAU LE THOUX A CURIS AU MONT D'OR 2
16150583	LE COURS D'EAU NUMÉRO 01 DE LA BÉLINIÈRE A ...
26405099	REAL MARTIN A PIGNANS 1

	LbStationMesureEauxSurface
142208	Le Majesq à Azur
468800	BOURBRE A CHAVANOZ
490122	EYGUES A ST-MAURICE/EYGUES - LES CIVARDIERES
645026	MOURACHONNE A PEGOMAS
1980639	BIEF D'ENFER A ST ETIENNE SUR REYSSOUZE
2024300	BUGEON A LA-CHAMBRE
2595740	La Véronne en aval de Riom-ès-Montagnes (Amont...)
6148279	TORRENSON A ST-CYR
10095956	AGNY A NIVOLAS-VERMELLE
13028690	SANSFOND A SAULON-LA-RUE
15703808	RUISSEAU LE THOUX A CURIS AU MONT D'OR
16150583	LE COURS D'EAU NUMÉRO 01 DE LA BÉLINIÈRE A CON...
26405099	REAL MARTIN A PIGNANS

Les labels sont différents mais ont l'air de correspondre à la même chose.

```
label_diff_pc_station =
merged_all[merged_all['LbStationMesureEauxSurface_pc'] != merged_all['LbStationMesureEauxSurface']]
label_diff_pc_station = label_diff_pc_station.loc[:, ['CdStationMesureEauxSurface', 'LbStationMesureEauxSurface_pc', 'LbStationMesureEauxSurface']]
label_diff_pc_station = label_diff_pc_station.drop_duplicates()
label_diff_pc_station

CdStationMesureEauxSurface      LbStationMesureEauxSurface_pc
\
6156                      5010000          Le Pharaon à St-Pardon
11396                     5015100          Le Charreau à St-Michel
```

77052	5116100	La Séoune à Montjoi
475147	6580578	AIGUE NOIRE A DOMESSIN
505611	6010000	OGNON A PESMES 1
...
24098987	1000602	COLOGNE À BUIRE COURCELLES (80)
24105377	1001131	HELPE MINEURE À GRAND FAYT (59)
26676374	4108492	LONG À DISSAY-SOUS-COURCILLON
26687107	4406063	LE BONSON A PERIGNEUX
42300999	6446330	BLUSSANS A BLUSSANS 1

	LbStationMesureEauxSurface
6156	Le Pharon à St-Pardon
11396	Le Charraud à St-Michel
77052	La Séoune à Montjoie
475147	RUISSEAU D'AIGUE NOIRE A DOMESSIN
505611	OGNON A PESMES
...	...
24098987	COLOGNE à BUIRE COURCELLES (80)
24105377	HELPE MINEURE à GRAND FAYT (59)
26676374	R LONG À DISSAY-SOUS-COURCILLON
26687107	TURLET
42300999	RUISSEAU DE BLUSSANS A BLUSSANS

[71 rows x 3 columns]

label_diff_pc_station.head(20)

CdStationMesureEauxSurface	\
6156	5010000
11396	5015100
77052	5116100
475147	6580578
505611	6010000
545763	6143950
597663	6135350
598125	6070460
611335	6144900
644341	6078500
668593	6106935
677174	6580563
700205	6055000
706232	6202100

723439	6065675
1082462	5015250
1629659	6085500
1667982	6047500
1692218	6580582
1719834	6167000

	LbStationMesureEauxSurface_pc \
6156	Le Pharaon à St-Pardon
11396	Le Charreau à St-Michel
77052	La Séoune à Montjoi
475147	AIGUE NOIRE A DOMESSIN
505611	OGNON A PESMES 1
545763	ROMANCHE A BOURG D'OISANS - LE PONT ROUGE 2
597663	PLANAY A MEGEVE 1
598125	NANT AILLON A AILLON-LE-VIEUX
611335	ROMANCHE A JARRIE 1
644341	TIER A BELMONT-TRAMONET 1
668593	DORNE A DORNAS 2
677174	BONNARD A SAINT-BERON 1
700205	BREVENNE A SAIN-BEL 1
706232	GAPEAU A SIGNES 1
723439	COPPY A MAXILLY-SUR-LEMAN 1
1082462	Le Charreau à Torsac
1629659	BIENNE A JEURRE 1
1667982	PETITE GROSNE A MACON 1
1692218	VIERAN A ANNECY 1
1719834	TECH A REYNES 1

	LbStationMesureEauxSurface
6156	Le Pharon à St-Pardon
11396	Le Charraud à St-Michel
77052	La Séoune à Montjoie
475147	RUISSEAU D'AIGUE NOIRE A DOMESSIN
505611	OGNON A PESMES
545763	ROMANCHE A BOURG D'OISANS - LE PONT ROUGE
597663	PLANAY A MEGEVE
598125	NANT D'AILLON A AILLON-LE-VIEUX
611335	ROMANCHE A JARRIE
644341	TIER A BELMONT-TRAMONET
668593	DORNE A DORNAS
677174	RUISSEAU DE BONNARD A ST-BERON
700205	BREVENNE A SAIN-BEL
706232	GAPEAU A SIGNES
723439	RUISSEAU DE COPPY A MAXILLY-SUR-LEMAN
1082462	Le Charraud à Torsac
1629659	BIENNE A JEURRE
1667982	PETITE GROSNE A MACON

1692218
1719834

VIERAN A MEYTHET
TECH A REYNES

afficher les données de 20 à 40
label_diff_pc_station[20:40]

	CdStationMesureEauxSurface \
1733672	6178800
2029742	6176130
2595740	5068920
2606716	5074920
3071988	6115080
3174351	6084360
3299210	6182120
3327085	6179550
3334156	6464800
3346565	6580960
3528228	6178050
3534277	6830180
3634331	6168200
3635010	6148850
3969301	3096650
5600749	6107760
5869307	6165700
5933441	6830800
6102244	6300056
6131780	6470900

	LbStationMesureEauxSurface_pc \
1733672	ORBIEL A LES-MARTYS
2029742	AUDE A LIMOUX 1
2595740	La Véronne en aval de Riom-ès-Montagnes (Amont...)
2606716	Le Gat-Mort à Villagrains
3071988	IBIE A LAGORCE 1
3174351	AIN A MESNOIS 1
3299210	HERAULT A PUECHABON 1
3327085	ARGENT DOUBLE A AZILLE 1
3334156	CLAUGE A LA-LOYE 1
3346565	GRESSE A VARCES-ALLIERES-ET-RISSET 3
3528228	SOUPEX A SOUILHE
3534277	RISSE A ST-JEOIRE 1
3634331	MASSANE A ARGELES-SUR-MER 1
3635010	SAVASSE A ROMANS-SUR-ISERE 2
3969301	LA CHEE A MERLAUT 1
5600749	DUNIERE A SILHAC 1
5869307	EZE A PERTUIS 3
5933441	MORGE A VALLIERES 2
6102244	MOSSON A MONTPELLIER
6131780	GROZONNE A NEUVILLEY 1

	LbStationMesureEauxSurface
1733672	ORBIEL A LES-MARTYRS
2029742	AUDE A LIMOUX
2595740	La Véronne en aval de Riom-ès-Montagnes (Amont...)
2606716	Le Gat Mort à Villagrains
3071988	IBIE A VALLON-PONT-D'ARC
3174351	AIN A MESNOIS
3299210	HERAULT A PUECHABON
3327085	ARGENT DOUBLE A AZILLE
3334156	CLAUGE A LA-LOYE
3346565	GRESSE A VARCES-ALLIERES-ET-RISSET
3528228	FRESQUEL A SOUILHE
3534277	RISSE A ST-JEOIRE-EN-FAUCIGNY 1
3634331	MASSANA A ARGELES-SUR-MER
3635010	SAVASSE A ROMANS-SUR-SERE 2
3969301	LA CHÉE A MERLAUT 1
5600749	DUNIERE A SILHAC
5869307	LEZE A PERTUIS 3
5933441	MORGE A VAL-DE-FIER
6102244	MOSSON A PONTPELLIER
6131780	GROZONNE A NEUVILLEY

40 à 60

label_diff_pc_station[40:60]

	CdStationMesureEauxSurface \
6175174	6041700
7416343	6820139
7422432	6580794
7550442	6820144
7718952	6580793
7837573	6436900
10399011	6084210
10627070	6070750
10660253	6041280
10674197	6830122
12499089	6491650
12500544	6440850
12996658	6440770
13050103	6440750
13071231	6440950
14008955	5211550
16150583	3250952
17055614	2077150
17063317	2106815
17620063	6028110

	LbStationMesureEauxSurface_pc \
6175174	BRENNE A SENS-SUR-SEILLE 1
7416343	GIER A L'HORME 1

7422432	JANON A ST-CHAMOND 1
7550442	MORNANTE A ST-CHAMOND 1
7718952	RICOLIN A ST-CHAMOND 1
7837573	CORCELLE A RIGNEY 1
10399011	DROUVENANT A BOISSIA 1
10627070	DADON A RUMILLY 1
10660253	LEMME A LAC DES ROUGES TRUITES 1
10674197	FILLIERE A FILLIERE 1
12499089	SEDAN A VILLEVIEUX 1
12500544	GRAVELLON A THERVAY 1
12996658	FONTAINE DE MAGNEY A SORNAY 1
13050103	FONTAINE DE DOUIS RU A MARNAY 1
13071231	VEZE A VITREUX 1
14008955	La Lèze à Monein
16150583	LE COURS D'EAU NUMÉRO 01 DE LA BÉLINIÈRE A CON...
17055614	LE RUPT DE MAD À RAMBUCOURT
17063317	LA SAÔNNELLE À FRÉBÉCOURT
17620063	GRANDS TERREAUX A SAONE 2

	LbStationMesureEauxSurface
6175174	BRENNE A SENS-SUR-SEILLE
7416343	GIER A L'HORME
7422432	JANON A ST-CHAMOND
7550442	MORNANTE A ST-CHAMOND
7718952	RICOLIN A ST-CHAMOND
7837573	CORCELLE A RIGNEY
10399011	DROUVENANT A BOISSIA
10627070	DADON A RUMILLY
10660253	LEMME A LAC DES ROUGES TRUITES
10674197	FILLIERE A THORENS-GLIERES 1
12499089	SEDAN A LARNAUD
12500544	GRAVELLON A THERVAY
12996658	RUISSEAU DE LA FONTAINE DE MAGNEY A SORNAY
13050103	FONTAINE DE DOUIS RU A MARNAY
13071231	RUISSEAU DE LA VEZE A VITREUX 1
14008955	Le Luzoué à Monein
16150583	LE COURS D'EAU NUMÉRO 01 DE LA BÉLINIÈRE A CON...
17055614	LE RUPT-DE-MAD À RAMBUCOURT
17063317	LA SAÔNNELLE À FRÉBÉCOURT
17620063	RUISSEAU DES GRANDS TERREAUX A SAONE

60 à la fin

label_diff_pc_station[60:]

	CdStationMesureEauxSurface	LbStationMesureEauxSurface_pc
\		
17816874	6028100	MARAIS A SAONE 1
18204892	6461520	BREVILLIERS A HERICOURT 2

22555554	6083710	LEMME A ENTRE DEUX MONTS 1
23433166	6068410	DORCHE A CHANAY 1
24028092	1002228	LA TERNOISE À TILLY CAPELLE (62)
24097129	1002222	LA RIVIERETTE À LE FAVRIL (59)
24098987	1000602	COLOGNE À BUIRE COURCELLES (80)
24105377	1001131	HELPE MINEURE À GRAND FAYT (59)
26676374	4108492	LONG À DISSAY-SOUS-COURCILLON
26687107	4406063	LE BONSON A PERIGNEUX
42300999	6446330	BLUSSANS A BLUSSANS 1

	LbStationMesureEauxSurface
17816874	RUISSEAU DES MARAIS A SAONE 1
18204892	RUISSEAU DE BREVILLIERS A HERICOURT 2
22555554	LEMME A ENTRE DEUX MONTS
23433166	LA DORCHE A CHANAY 1
24028092	LA TERNOISE A TILLY CAPELLE (62)
24097129	LA RIVIERETTE A LE FAVRIL (59)
24098987	COLOGNE à BUIRE COURCELLES (80)
24105377	HELPE MINEURE à GRAND FAYT (59)
26676374	R LONG À DISSAY-SOUS-COURCILLON
26687107	TURLET
42300999	RUISSEAU DE BLUSSANS A BLUSSANS

Les différences entre les labels des stations semblent être uniquement dues à des fautes de frappe ou à des abréviations différentes.

Pour vérifier, nous avons localisé les stations sur une carte afin de confirmer leur correspondance :

- 16580582 : vérifié, correspond bien.
- 6115080 : vérifié, correspond bien.
- 6178050 : vérifié, correspond bien.
- 6830800 : vérifié, correspond bien.
- 6830122 : vérifié, correspond bien.

Cependant, pour le code 4406063, nous n'avons pas pu vérifier, nous avons donc décidé de la supprimer par précaution.

Appliquer le nettoyage aux différents dataframes

```
# On applique à df_stations, au dataframe de physicochimique et au
# dataframe d'hydrobiologie la préparation
```

```

df_stations.rename(columns={'CdStationMesureEauxSurface': 'station'},
inplace=True)
df_stations = clean_station_column(df_stations, 'station')
df_pc_pivot_saison_mean =
clean_station_column(df_pc_pivot_saison_mean, 'station')
df_pc_pivot_saison_median =
clean_station_column(df_pc_pivot_saison_median, 'station')
df_hydrobio_lag_1_month_median =
clean_station_column(df_hydrobio_lag_1_month_median, 'station')
df_hydrobio_lag_3_month_median =
clean_station_column(df_hydrobio_lag_3_month_median, 'station')
df_hydrobio_lag_6_month_median =
clean_station_column(df_hydrobio_lag_6_month_median, 'station')
df_hydrobio_lag_1_month_mean =
clean_station_column(df_hydrobio_lag_1_month_mean, 'station')
df_hydrobio_lag_3_month_mean =
clean_station_column(df_hydrobio_lag_3_month_mean, 'station')
df_hydrobio_lag_6_month_mean =
clean_station_column(df_hydrobio_lag_6_month_mean, 'station')
# supprimer les lignes avec 4406063
df_stations = df_stations[df_stations['station'] != 4406063]
df_pc_pivot_saison_mean =
df_pc_pivot_saison_mean[df_pc_pivot_saison_mean['station'] != 4406063]
df_pc_pivot_saison_median =
df_pc_pivot_saison_median[df_pc_pivot_saison_median['station'] !=
4406063]
df_hydrobio_lag_1_month_median =
df_hydrobio_lag_1_month_median[df_hydrobio_lag_1_month_median['station
'] != 4406063]
df_hydrobio_lag_3_month_median =
df_hydrobio_lag_3_month_median[df_hydrobio_lag_3_month_median['station
'] != 4406063]
df_hydrobio_lag_6_month_median =
df_hydrobio_lag_6_month_median[df_hydrobio_lag_6_month_median['station
'] != 4406063]
df_hydrobio_lag_1_month_mean =
df_hydrobio_lag_1_month_mean[df_hydrobio_lag_1_month_mean['station'] !=
= 4406063]
df_hydrobio_lag_3_month_mean =
df_hydrobio_lag_3_month_mean[df_hydrobio_lag_3_month_mean['station'] !=
= 4406063]
df_hydrobio_lag_6_month_mean =
df_hydrobio_lag_6_month_mean[df_hydrobio_lag_6_month_mean['station'] !=
= 4406063]

```

Aggrégation des données physicochimiques et hydrobiologiques

Nous avons fusionné les données physicochimiques et hydrobiologiques (avec un décalage d'un mois) agrégées par médiane dans un seul dataset, en utilisant les colonnes communes `station`, `année`, et `saison`.

À partir de cette étape, toutes les analyses seront effectuées sur ce dataset fusionné `df_pc_median_bio_median_1_month`.

```
df_pc_median_bio_median_1_month = pd.merge(df_pc_pivot_saison_median,  
df_hydrobio_lag_1_month_median, on=['station', 'année', 'saison'],  
how='inner')
```

```
df_pc_median_bio_median_1_month.head(5)
```

	station	année	saison	Ammonium - Eau	Azote Kjeldahl - Eau	\
0	5001800	2007	Été	0.04		1.0
1	5001800	2008	Été	0.04		1.0
2	5001800	2009	Été	0.02		1.0
3	5004000	2007	Été	0.04		1.0
4	5004000	2008	Été	0.03		1.0

	Carbone Organique - Eau	Conductivité à 25°C - Eau	\
0	3.5	765.0	
1	2.7	765.0	
2	2.2	736.0	
3	2.8	683.0	
4	2.1	689.0	

	Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	Diuron - Eau	\
0		0.5	
0.01			
1		0.6	
NaN			
2		1.0	
0.02			
3		0.5	
0.01			
4		0.5	
NaN			

	Matières en suspension - Eau	Nitrates - Eau	Nitrites - Eau	\
0	3.0	36.2	0.09	
1	4.0	40.3	0.07	
2	6.0	37.9	0.07	
3	3.0	26.4	0.06	
4	3.0	32.3	0.06	

	Orthophosphates (P04) - Eau	Oxygène dissous - Eau	Phosphore total
--	-----------------------------	-----------------------	-----------------

```
- Eau \
0          0.05          8.1
0.05
1          0.05          8.3
0.11
2          0.05          7.4
0.07
3          0.05          7.3
0.06
4          0.05          6.1
0.09
```

```
Potentiel en Hydrogène (pH) - Eau   Taux de saturation en oxygène -
Eau \
0          7.9
89.0
1          7.9
90.0
2          8.0
83.0
3          7.7
77.0
4          7.6
63.0
```

```
Température de l'Eau - Eau   Turbidité Formazine Néphéломétrique -
Eau \
0          18.6
NaN
1          20.5
NaN
2          20.6
NaN
3          17.8
NaN
4          17.3
NaN
```

```
I2M2
0  0.3004
1  0.3848
2  0.5756
3  0.3992
4  0.3383
```

```
df_pc_median_bio_median_1_month.shape
(37966, 20)
```

```

# Statistiques descriptives
desc_stats_saison = df_pc_median_bio_median_1_month.groupby('saison')
['I2M2'].describe()
desc_stats_saison =
desc_stats_saison.drop(columns=['count']).transpose()
fig = go.Figure()
for saison in desc_stats_saison.columns:
    fig.add_trace(go.Bar(x=desc_stats_saison.index,
y=desc_stats_saison[saison],name=saison))
for etat, valeur in seuils_I2M2.items():
    fig.add_shape(type="line", x0=-0.5, x1=len(desc_stats)-0.5,y0=valeur, yl=valeur, line=dict(color="black", width=1,
dash="dash"), name=etat)
    fig.add_annotation(x=len(desc_stats)-0.5, y=valeur, text=etat,
showarrow=False, font=dict(size=10, color="black"), xanchor="left" )
fig.update_layout(height=500, width=700, font_size=10,
title="Statistiques descriptives de I2M2 par Saison avec un lag de 1 mois",
xaxis_title="Statistique",
yaxis_title="Valeur",barmode='group')
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
{"name": "Automne", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.6091704246949807, 0.2389319089190886, 0, 0.459, 0.6678, 0.79745, 0.9909]}, {"name": "Hiver", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.6166104118993135, 0.15414801991904364, 0, 0.52925, 0.6395, 0.7304999999999999, 0.955]}, {"name": "Printemps", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.45310416316316315, 0.23171656266004786, 0, 0.2689, 0.469, 0.64, 0.999]}, {"name": "Été", "type": "bar", "x": ["mean", "std", "min", "25%", "50%", "75%", "max"], "y": [0.5505881598624405, 0.23426350920432412, 0, 0.3888, 0.593, 0.738, 1]}], "layout": {"annotations": [{"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Très bon", "x": 6.5, "xanchor": "left", "y": 0.665}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Bon", "x": 6.5, "xanchor": "left", "y": 0.443}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Moyen", "x": 6.5, "xanchor": "left", "y": 0.295}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Médiocre", "x": 6.5, "xanchor": "left", "y": 0.148}, {"font": {"color": "black", "size": 10}, "showarrow": false, "text": "Mauvais", "x": 6.5, "xanchor": "left", "y": 0}], "barmode": "group", "font": {"size": 10}, "height": 500, "shapes": [{"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Très bon"}, {"line": {"color": "black", "dash": "line", "x0": -0.5, "x1": 6.5, "y0": 0.665, "y1": 0.665}, {"line": {"color": "black", "dash": "dash", "width": 1}, "name": "Bon"}, {"line": {"color": "black", "dash": "line", "x0": -0.5, "x1": 6.5, "y0": 0.443, "y1": 0.443}]}]

```

```

    {"color": "black", "dash": "dash", "width": 1}, {"name": "Moyen", "type": "line"}, {"x0": -0.5, "x1": 6.5, "y0": 0.295, "y1": 0.295}, {"line": {"color": "black", "dash": "dash", "width": 1}, {"name": "Médiocre", "type": "line"}, {"x0": -0.5, "x1": 6.5, "y0": 0.148, "y1": 0.148}, {"line": {"color": "black", "dash": "dash", "width": 1}, {"name": "Mauvais", "type": "line"}, {"x0": -0.5, "x1": 6.5, "y0": 0, "y1": 0}], "template": {"data": [{"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, {"choropleth": [{"outlinewidth": 0, "ticks": ""}, {"type": "choropleth"}], "contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], {"type": "contour"}], "contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, {"type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, {"type": "contourcarpet"}], {"type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, {"type": "contour"}], [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], {"type": "heatmapgl"}], "histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}], "histogram2d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, {"type": "histogram2d"}], [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], {"type": "histogram2d"}], "histogram2dcontour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, {"type": "histogram2dcontour"}]}]
  
```

```

[[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],

[1, "#f0f921]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"autoMargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}, "type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}}, {"type": "surface"}], [{"color": "#0d0887"}, {"color": "#46039f"}, {"color": "#7201a8"}, {"color": "#9c179e"}, {"color": "#bd3786"}, {"color": "#d8576b"}, {"color": "#ed7953"}, {"color": "#fb9f3a"}, {"color": "#fdca26"}], [{"color": "#EBF0F8"}, {"color": "#C8D4E3"}], [{"color": "#2a3f5f"}], [{"arrowColor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1, "autoTypeNumbers": "strict", "colorAxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorScale": {"diverging": [{"color": "#8e0152"}, {"color": "#c51b7d"}, {"color": "#de77ae"}, {"color": "#f1b6da"}, {"color": "#fde0ef"}, {"color": "#f7f7f7"}, {"color": "#e6f5d0"}, {"color": "#b8e186"}, {"color": "#7fbcc41"}, {"color": "#4d9221"}, {"color": "#276419"}], "sequential": [{"color": "#0d0887"}, {"color": "#46039f"}, {"color": "#7201a8"}, {"color": "#9c179e"}, {"color": "#bd3786"}, {"color": "#d8576b"}, {"color": "#ed7953"}, {"color": "#fb9f3a"}, {"color": "#fdca26"}], [{"color": "#f0f921"}]]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [{"color": "#8e0152"}, {"color": "#c51b7d"}, {"color": "#de77ae"}, {"color": "#f1b6da"}, {"color": "#fde0ef"}, {"color": "#f7f7f7"}, {"color": "#e6f5d0"}, {"color": "#b8e186"}, {"color": "#7fbcc41"}, {"color": "#4d9221"}, {"color": "#276419"}], "sequential": [{"color": "#0d0887"}, {"color": "#46039f"}, {"color": "#7201a8"}, {"color": "#9c179e"}, {"color": "#bd3786"}, {"color": "#d8576b"}, {"color": "#ed7953"}, {"color": "#fb9f3a"}, {"color": "#fdca26"}], [{"color": "#f0f921"}]}], "type": "sequentialminus"}], [{"color": "#0d0887"}, {"color": "#46039f"}]

```

```

[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "colorway": [
  "#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692",
  "#B6E880", "#FF97FF", "#FECB52"], "font": {"color": "#2a3f5f"}, "geo": {
    "bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}}, "title": {"text": "Statistiques descriptives de l'I2M2 par Saison avec un lag de 1 mois"}, "width": 700, "xaxis": {"title": {"text": "Statistique"}}, "yaxis": {"title": {"text": "Valeur"}}}}

```

Après la fusion des datasets, les statistiques descriptives de l'I2M2 restent assez similaires. La fusion n'a donc pas altéré les tendances principales de l'I2M2.

Aggrégation du dataset avec les données des stations et des hydroécorégions

Nous allons maintenant fusionner notre dataset avec celui des stations afin de récupérer les coordonnées géographiques des stations et déterminer leur appartenance aux hydroécorégions.

Ensuite, nous vérifierons le nombre de stations restantes dans les hydroécorégions après la fusion de tous les datasets. Cette étape de fusion a réduit le nombre total de stations en raison des différences dans les codes de stations, qui n'étaient pas toujours compatibles pour faire une jointure.

```
# Fusion avec le dataset des stations pour récupérer les coordonnées
# des stations
df_pc_median_bio_median_1_month_with_coords = pd.merge(
    df_pc_median_bio_median_1_month,
    df_stations[['station', 'CoordXStationMesureEauxSurface',
    'CoordYStationMesureEauxSurface']],
    on='station',
    how='inner'
)
```

Observation de la répartition des stations dans les hydroécorégions

```
df_to_count = df_pc_median_bio_median_1_month_with_coords.copy()

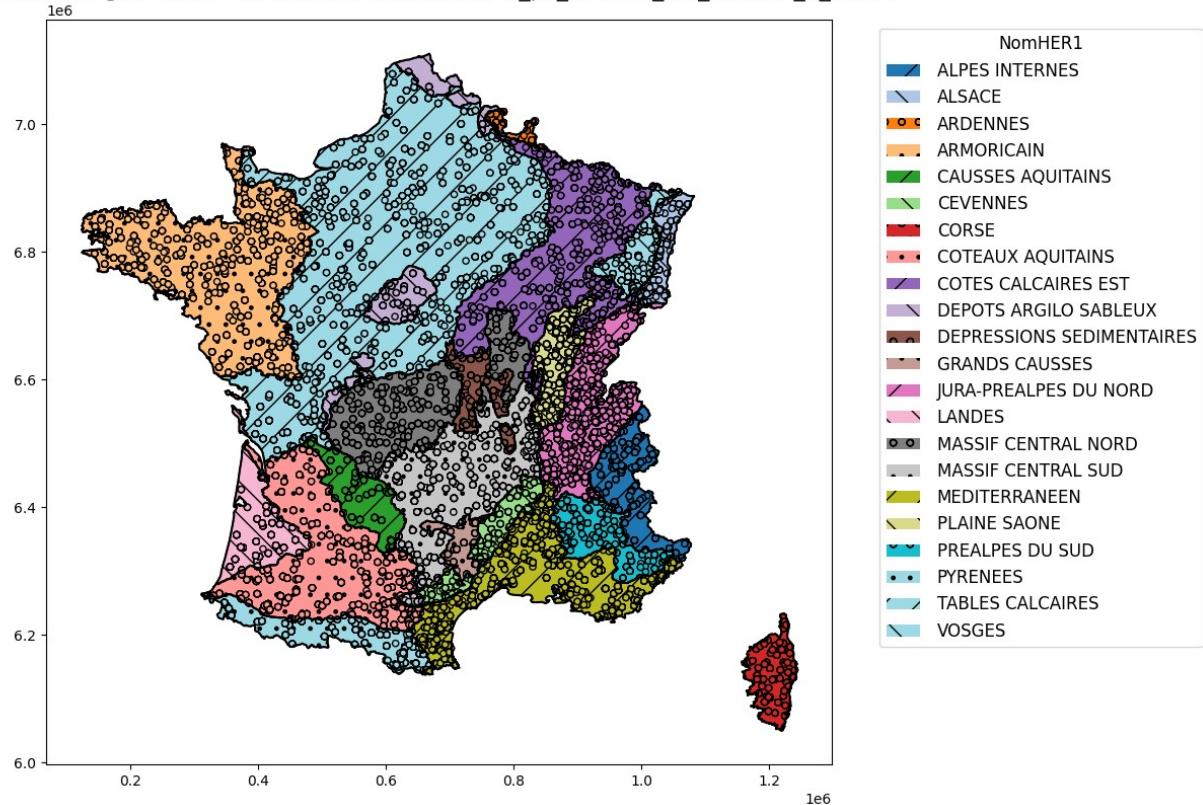
# Représentation des stations sur la carte des hydroécorégions
crs_lambert =
'PROJCS["RGF_1993_Lambert_93",GEOGCS["GCS_RGF_1993",DATUM["D_RGF_1993",
SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],
UNIT["Degree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],
PARAMETER["False_Easting",700000.0],PARAMETER["False_Northing",660000.0],
PARAMETER["Central_Meridian",3.0],PARAMETER["Standard_Parallel_1",49.0],
PARAMETER["Standard_Parallel_2",44.0],PARAMETER["Latitude_of_Origin",46.5],
UNIT["Meter",1.0]]'
x_col = 'CoordXStationMesureEauxSurface'
y_col = 'CoordYStationMesureEauxSurface'
carto_i2m2 = gpd.GeoDataFrame(df_to_count, crs=crs_lambert ,
                               geometry =
gpd.GeoSeries(df_to_count.agg(lambda
x:geom.Point(x.loc[x_col],x.loc[y_col]) ,axis=1)))
# Récupération des hydroécorégions pour les stations
HER_stations=carto_i2m2.sjoin(df_hydroregions.to_crs(crs_lambert),predicate='within').to_crs(crs_lambert)
fig, ax = plt.subplots(1, 1, figsize=(10, 30))
colors = plt.colormaps['tab20']
hatches = ['/', '\\", 'o', '.']
legend_elements = []
color_mapping = {}
for i, (name, region) in
enumerate(df_hydroregions.groupby('NomHER1')):
    region = region.to_crs(crs_lambert)
    patch = region.plot(ax=ax, color=colors(i), hatch=hatches[i %
len(hatches)], edgecolor='black', label=name)
    legend_elements.append(Patch(facecolor=colors(i), hatch=hatches[i %
len(hatches)], label=name))
    color_mapping[name] = colors(i)
```

```

station_colors = HER_stations['NomHER1'].map(color_mapping)
HER_stations.plot(ax=ax, color=station_colors, markersize=20,
edgecolor='black')
HER_lambert = df_hydroregions.to_crs(crs_lambert)
HER_lambert.boundary.plot(ax=ax, color='black')
ax.legend(handles=legend_elements, title='NomHER1',
bbox_to_anchor=(1.05, 1), loc='upper left', fontsize='large',
title_fontsize='large', handletextpad=1)
ax.set_title('Hydroecoregions avec les stations du dataset
df_pc_median_bio_median_1_month', fontsize=16)
plt.show()

```

Hydroecoregions avec les stations du dataset df_pc_median_bio_median_1_month



```

print(f"Nombre de doublons dans df_stations :
{df_stations.duplicated(subset='station').sum()}")
print(f"Nombre de doublons dans
df_pc_median_bio_median_1_month_with_coords :
{df_to_count.duplicated(subset='station').sum()}")

```

Nombre de doublons dans df_stations : 7
Nombre de doublons dans df_pc_median_bio_median_1_month_with_coords : 18909

```

df_stations.drop_duplicates(subset='station', inplace=True)
df_to_count.drop_duplicates(subset='station', inplace=True)

```

```

# Récupération des hydroécorégions pour les stations dans df_stations
carto_i2m2_1 = gpd.GeoDataFrame(df_stations, crs=crs_lambert, geometry=gpd.GeoSeries(df_stations.agg(lambda
x:geom.Point(x.loc['CoordXStationMesureEauxSurface'],x.loc['CoordYStationMesureEauxSurface']) ,axis=1)))
HER_stations_1=carto_i2m2_1.sjoin(df_hydroregions.to_crs(crs_lambert), predicate='within').to_crs(crs_lambert)
# Récupération des hydroécorégions pour les stations dans
df_pc_median_bio_median_1_month_with_coords
carto_i2m2_2 = gpd.GeoDataFrame(df_to_count, crs=crs_lambert, geometry=gpd.GeoSeries(df_to_count.agg(lambda
x:geom.Point(x.loc['CoordXStationMesureEauxSurface'],x.loc['CoordYStationMesureEauxSurface']) ,axis=1)))
HER_stations_2=carto_i2m2_2.sjoin(df_hydroregions.to_crs(crs_lambert), predicate='within').to_crs(crs_lambert)

print(f"Nombre de doublons après jointure spatiale dans HER_stations_1 : {HER_stations_1.duplicated(subset='station').sum()}")
print(f"Nombre de doublons après jointure spatiale dans HER_stations : {HER_stations_2.duplicated(subset='station').sum()}")

Nombre de doublons après jointure spatiale dans HER_stations_1 : 0
Nombre de doublons après jointure spatiale dans HER_stations : 0

# Nombre de stations par région hydroécologique dans df_stations et
# dans df_pc_median_bio_median_1_month
# df_stations
stations_count_by_region_1 =
HER_stations_1.groupby('NomHER1').size().reset_index(name='count')
# df_pc_median_bio_median_1_month
stations_count_by_region_2 =
HER_stations_2.groupby('NomHER1').size().reset_index(name='count')
# comparaison du nombre de stations entre les deux dataframes
comparison = pd.merge(stations_count_by_region_1,
stations_count_by_region_2, on='NomHER1', how='outer',
suffixes=('_df_stations', '_df_pc_median'))
print(comparison)

```

	NomHER1	count_df_stations	count_df_pc_median
0	ALPES INTERNES	156	84
1	ALSACE	352	97
2	ARDENNES	63	15
3	ARMORICAIN	445	249
4	CAUSSES AQUITAINS	47	31
5	CEVENNES	106	81
6	CORSE	63	47
7	COTEAUX AQUITAINS	265	188
8	COTES CALCAIRES EST	1008	269
9	DEPOTS ARGILEO SABLEUX	47	35
10	DEPRESSIONS SEDIMENTAIRES	63	43

11	GRANDS CAUSSES	25	18
12	JURA-PREALPES DU NORD	628	390
13	LANDES	38	30
14	MASSIF CENTRAL NORD	221	137
15	MASSIF CENTRAL SUD	264	215
16	MEDITERRANEEN	571	376
17	PLAINE SAONE	222	170
18	PREALPES DU SUD	167	85
19	PYRENEES	116	84
20	TABLES CALCAIRES	1360	431
21	VOSGES	313	64

```
# Affichage du nombre de stations par hydroécorégion dans les 2
dataframes
comparison_sorted = comparison.sort_values('count_df_pc_median',
ascending=True)
fig = go.Figure(data=[
    go.Bar(x=comparison_sorted['NomHER1'],
y=comparison_sorted['count_df_stations'], name='df_stations'),
    go.Bar(x=comparison_sorted['NomHER1'],
y=comparison_sorted['count_df_pc_median'], name='df_pc_median_bio_median_1_month')
])
fig.update_layout(title="Comparaison du nombre de stations par
hydroécorégion", xaxis_title="Hydroécorégions", yaxis_title="Nombre de
stations", barmode='group', xaxis=dict(tickangle=90))
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
{"name": "df_stations", "type": "bar", "x": ["ARDENNES", "GRANDS
CAUSSES", "LANDES", "CAUSSES AQUITAINS", "DEPOTS ARGIGO
SABLEUX", "DEPRESSIONS
SEDIMENTAIRES", "CORSE", "VOSGES", "CEVENNES", "PYRENEES", "ALPES
INTERNES", "PREALPES DU SUD", "ALSACE", "MASSIF CENTRAL NORD", "PLAINE
SAONE", "COTEAUX AQUITAINS", "MASSIF CENTRAL SUD", "ARMORICAIN", "COTES
CALCAIRES EST", "MEDITERRANEEN", "JURA-PREALPES DU NORD", "TABLES
CALCAIRES"], "y": [63, 25, 38, 47, 47, 63, 63, 313, 106, 116, 156, 167, 352, 221, 222, 265, 264, 445, 1008
, 571, 628, 1360]}, {"name": "df_pc_median_bio_median_1_month", "type": "bar", "x": ["ARDENNES", "GRANDS CAUSSES", "LANDES", "CAUSSES AQUITAINS", "DEPOTS
ARGIGO SABLEUX", "DEPRESSIONS
SEDIMENTAIRES", "CORSE", "VOSGES", "CEVENNES", "PYRENEES", "ALPES
INTERNES", "PREALPES DU SUD", "ALSACE", "MASSIF CENTRAL NORD", "PLAINE
SAONE", "COTEAUX AQUITAINS", "MASSIF CENTRAL SUD", "ARMORICAIN", "COTES
CALCAIRES EST", "MEDITERRANEEN", "JURA-PREALPES DU NORD", "TABLES
CALCAIRES"], "y": [15, 18, 30, 31, 35, 43, 47, 64, 81, 84, 84, 85, 97, 137, 170, 188, 215, 249, 269, 376, 39
0, 431]}, {"layout": {"barmode": "group", "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": "
```

```

    {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": [{"fillmode": "overlay", "size": 10, "solidity": 0.2}], "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": [{"fillmode": "overlay", "size": 10, "solidity": 0.2}], "type": "barpolar"}]}, {"carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, {"choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}]}, {"contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "type": "contour"}]}, {"contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet"}]}, {"heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "type": "heatmap"}]}, {"heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "type": "heatmapgl"}]}, {"histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}]}, {"histogram2d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "type": "histogram2d"}]}, {"histogram2dcontour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "type": "histogram2dcontour"}]}, {"mesh3d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "mesh3d"}]}, {"parcoords": [{"line": {"color": "#2a3f5f", "width": 0.5}, "pattern": [{"fillmode": "overlay", "size": 10, "solidity": 0.2}], "type": "bar"}]}]

```

```

  {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "parcoords"}], "pie": [{"autoMargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}, "type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], {"type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], {"sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], {"sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]]}, {"colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECB52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlake": true}]}]}]}]
```

```

    s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
    {"align":"left"},"hovermode":"closest","mapbox":
    {"style":"light"},"paper_bgcolor":"white","plot_bgcolor": "#E5ECF6","po
    lar": {"angularaxis":
    {"gridcolor": "white","linecolor": "white","ticks": ""}, "bgcolor": "#E5ECF
    6","radialaxis":
    {"gridcolor": "white","linecolor": "white","ticks": ""}}, "scene":
    {"xaxis":
    {"backgroundcolor": "#E5ECF6","gridcolor": "white","gridwidth": 2,"lineco
    lor": "white","showbackground": true,"ticks": "", "zerolinecolor": "white"
    }, "yaxis":
    {"backgroundcolor": "#E5ECF6","gridcolor": "white","gridwidth": 2,"lineco
    lor": "white","showbackground": true,"ticks": "", "zerolinecolor": "white"
    }, "zaxis":
    {"backgroundcolor": "#E5ECF6","gridcolor": "white","gridwidth": 2,"lineco
    lor": "white","showbackground": true,"ticks": "", "zerolinecolor": "white"
    }, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis":
    {"gridcolor": "white","linecolor": "white","ticks": ""}, "baxis":
    {"gridcolor": "white","linecolor": "white","ticks": ""}, "bgcolor": "#E5ECF
    6", "caxis":
    {"gridcolor": "white","linecolor": "white","ticks": ""}}, "title":
    {"x": 5.0e-2}, "xaxis":
    {"automargin": true,"gridcolor": "white","linecolor": "white","ticks": "", "title":
    {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis":
    {"automargin": true,"gridcolor": "white","linecolor": "white","ticks": "", "title":
    {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}, "title":
    {"text": "Comparaison du nombre de stations par
    hydroécorégion"}, "xaxis": {"tickangle": 90, "title":
    {"text": "Hydroécorégions"}}, "yaxis": {"title": {"text": "Nombre de
    stations"}}}}

```

Certaines hydrorégions sont très sous représentées, et les plus représentées ont au maximum 430 stations ce qui reste peu.

```

# Calcul de la perte
comparison['percentage_loss'] = ((comparison['count_df_stations'] -
comparison['count_df_pc_median'])/ comparison['count_df_stations']) *
100
comparison['percentage_loss'] =
comparison['percentage_loss'].fillna(0)
comparison.sort_values('percentage_loss', inplace=True)

# Affichage du pourcentage de perte du nombre de stations par
hydroécorégion
fig = go.Figure(data=[go.Bar(x=comparison['NomHER1'],
y=comparison['percentage_loss'], marker_color='orange',
name='Pourcentage de perte')])

```

```

fig.update_layout(title="Pourcentage de perte des stations par hydroécorégion", xaxis_title="Hydroécorégions", yaxis_title="Pourcentage de perte (%)", xaxis=dict(tickangle=90))
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [{"marker": {"color": "orange"}, "name": "Pourcentage de perte", "type": "bar", "x": ["MASSIF CENTRAL SUD", "LANDES", "PLAINE SAONE", "CEVENNES", "CORSE", "DEPOTS ARGILLO SABLEUX", "PYRENEES", "GRANDS CAUSSES", "COTEAUX AQUITAINS", "DEPRESSIONS SEDIMENTAIRES", "CAUSSES AQUITAINS", "MEDITERRANEEN", "JURA-PREALPES DU NORD", "MASSIF CENTRAL NORD", "ARMORICAIN", "ALPES INTERNES", "PREALPES DU SUD", "TABLES CALCAIRES", "ALSACE", "COTES CALCAIRES EST", "ARDENNES", "VOSGES"], "y": [18.560606060606062, 21.052631578947366, 23.423423423423422, 23.58490566037736, 25.396825396825395, 25.53191489361702, 27.586206896551722, 28.00000000004, 29.056603773584904, 31.746031746031743, 34.04255319148936, 34.15061295971979, 37.898089171974526, 38.009049773755656, 44.04494382022472, 46.15384615384615, 49.101796407185624, 68.30882352941177, 72.44318181818183, 73.31349206349206, 76.19047619047619, 79.55271565495208]}], "layout": {"template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "bar"}}, {"barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "barpolar"}]}, {"carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "contour"}], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"]]]}]}]}]
```

```

[0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"],
[0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"],
[1, "#f0f921"], "type": "heatmapgl"}, "histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}],
"histogram2d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],

[1, "#f0f921], "type": "histogram2d"}], "histogram2dcontour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],

[1, "#f0f921], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"automargin": true, "type": "pie"}], "scatter": [{"fillpattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlinewidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlinewidth": 0, "ticks": ""}}, {"colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"],
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],

[1, "#f0f921], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationdefaults": {

```

```

{"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbcc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}, "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECB52"}, "font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}], "title": {"text": "Pourcentage de perte des stations par hydroécorégion"}, "xaxis": {"tickangle": 90}, "title": "
```

```

{"text": "Hydroécorégions"}, "yaxis": {"title": {"text": "Pourcentage de perte (%)"}}

# Calcul du nombre de stations perdues au total
total_stations_initial = comparison['count_df_stations'].sum()
total_stations_final = comparison['count_df_pc_median'].sum()
total_stations_perdue = total_stations_initial - total_stations_final
print(f"Nombre total de stations perdues : {total_stations_perdue}")
print(f"Pourcentage de stations perdues : {round((total_stations_perdue / total_stations_initial) * 100, 2)}%")

Nombre total de stations perdues : 3401
Pourcentage de stations perdues : 52.0%

```

Suite à la fusion des différents datasets, nous constatons une perte significative de stations par rapport au dataset initial. Au total, plus de 3000 stations, soit environ 50 % des stations, ont été perdues.

La distribution des pertes par hydroécorégion montre que certaines régions, comme les Vosges et les Ardennes, ont perdu une proportion importante de leurs stations, tandis que d'autres, comme le Massif Central Sud et la Plaine de la Saône, en ont conservé une plus grande partie, avec cependant environ 20% de perte.

Ces disparités régionales pourraient influencer les résultats de nos analyses.

Analyse des corrélations entre les caractéristiques

```

df_correl = df_pc_median_bio_median_1_month_with_coords.copy()

# Ajout des hydroécorégions pour voir si des caractéristiques y sont corrélées
carto_i2m2_correl = gpd.GeoDataFrame(df_correl, crs=crs_lambert,
geometry = gpd.GeoSeries(df_correl.agg(lambda x:geom.Point(x.loc['CoordXStationMesureEauxSurface'],x.loc['CoordYStationMesureEauxSurface']) ,axis=1)))
HER_stations_correl=carto_i2m2_correl.sjoin(df_hydroregions.to_crs(crs_lambert), predicate='within').to_crs(crs_lambert)
df_correl = HER_stations_correl.drop(columns=['geometry',
'index_right', 'NomHER1', 'gid'])

df_correl['saison'] = df_correl['saison'].factorize()[0]

# Mapping des colonnes pour pouvoir afficher la matrice de corrélation correctement
column_mapping = {
    'Ammonium - Eau': 'NH4',
    'Azote Kjeldahl - Eau': 'NKjeldahl',
    'Carbone Organique - Eau': 'CO',
    'Conductivité à 25°C - Eau': 'Cond25',
    'Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau': 'DB05',
    'Matières en suspension - Eau': 'MES',
}

```

```

'Nitrates - Eau': 'N03',
'Nitrites - Eau': 'N02',
'Orthophosphates (P04) - Eau': 'P04',
'Oxygène dissous - Eau': 'O2',
'Phosphore total - Eau': 'Ptot',
'Potentiel en Hydrogène (pH) - Eau': 'pH',
'Taux de saturation en oxygène - Eau': 'O2_sat',
'Température de l\'Eau - Eau': 'TempEau',
'Turbidité Formazine Néphélométrique - Eau': 'Turbidité',
'I2M2': 'I2M2',
'CoordXStationMesureEauxSurface': 'CoordX',
'CoordYStationMesureEauxSurface': 'CoordY'
}
df_correl = df_correl.rename(columns=column_mapping,
index=column_mapping)

# Création de la matrice de corrélation
correlation_matrix = df_correl.corr()
np.fill_diagonal(correlation_matrix.values, np.nan)
fig = px.imshow(correlation_matrix, text_auto=".1f",
color_continuous_scale='RdBu', title="Matrice de corrélation",
labels=dict(x="Caractéristiques", y="Caractéristiques",
color="Corrélation"), zmin=-1, zmax=1)
fig.update_layout(xaxis=dict(tickangle=45), autosize=True,
title_x=0.5, width=800, height=800)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
{"coloraxis": "coloraxis", "hovertemplate": "Caractéristiques: %
{x}<br>Caractéristiques: %{y}<br>Corrélation:
%{z}<extra></extra>", "name": "0", "texttemplate": "%
{z:.1f}", "type": "heatmap", "x": [
"station", "année", "saison", "NH4", "NKjeldahl", "C0", "Cond25", "DB05", "Di
uron -
Eau", "MES", "N03", "N02", "P04", "O2", "Ptot", "pH", "O2_sat", "TempEau", "Turb
idité", "I2M2", "CoordX", "CoordY", "CdHER1"], "xaxis": "x", "y": [
"station", "année", "saison", "NH4", "NKjeldahl", "C0", "Cond25", "DB05", "Di
uron -
Eau", "MES", "N03", "N02", "P04", "O2", "Ptot", "pH", "O2_sat", "TempEau", "Turb
idité", "I2M2", "CoordX", "CoordY", "CdHER1"], "yaxis": "y", "z": [[null, -
0.1745185670136645, 0.1827609237498206, 9.979277267703446e-
3, 0.11309580981749234, -0.28084443831181827, 2.1515754022741935e-2, -
0.18645096600808575, 0.14367627838279415, -1.2429569091361557e-2, -
0.3071766804640504, -0.10045931987772906, -
0.11876775074832747, 0.15155170179490057, -
0.130858540912874, 0.2538711026953098, 0.17978735434939427, -
6.785579219915185e-2, 1.1136665141270262e-2, -2.673347148426576e-
2, 0.2705620863637573, -0.6547384167787116, -0.29062096521357356], [-
0.1745185670136645, null, 6.408054023325764e-2, -8.002883466841133e-2, -
0.3863783409134091, 4.4689094471263374e-2, -2.2436171272371075e-2, -

```

5.506763147227038e-2, -0.24246494951667572, -1.0731168454959012e-
 2, 0.11071139941549857, -3.505062765102709e-2, 3.956082925739585e-
 3, 5.2340310851350755e-2, -2.0791058855023643e-2, -5.441375162084245e-2, -
 1.5608130072435064e-2, -8.099300725319886e-2, -4.593800458569556e-
 4, 2.1485586714586398e-2, -
 0.14482406000075418, 0.1902256295897464, 9.177707540627393e-2],
 [0.1827609237498206, 6.408054023325764e-2, null, 1.3291926080006278e-2, -
 1.866887291441953e-2, -0.14205911906483717, 9.112392165944976e-2, -
 4.852039914068734e-3, 2.280303547527108e-3, -2.479816749290803e-2, -
 6.13814926769179e-2, -6.8032022252419e-2, -
 0.1138859339718653, 0.5201343920288073, -
 0.11879376428796899, 0.18847556367509205, 0.1528367448304904, -
 0.7193731203252456, -2.7079224179623336e-2, -3.198067863083505e-
 2, 0.15196903802474168, -4.88457448222347e-2, -0.18196969216223963],
 [9.979277267703446e-3, -8.002883466841133e-2, 1.3291926080006278e-
 2, null, 0.5237581675681854, 0.1674859031551699, 0.15356562992058065, 0.311
 2017580295911, 0.1143491915729628, 7.548439434283552e-
 2, 2.4138230288449414e-2, 0.3700914093188439, 0.27881526060368567, -
 0.22042579963008363, 0.3902451915423873, -4.560653079591508e-2, -
 0.1732394296111281, 3.534490943430985e-2, 8.147978558553504e-2, -
 0.22430368544369425, 2.6144262364923634e-2, 1.2878725817232569e-
 2, 2.9657852023864163e-2], [0.11309580981749234, -0.3863783409134091, -
 1.866887291441953e-
 2, 0.5237581675681854, null, 0.28118293671202005, 0.11432632371779464, 0.32
 783014348784584, 0.1258806309722324, 0.2150750024954651, -
 4.673410496145254e-2, 0.2832034213062188, 0.2303081142515371, -
 0.20397424701090383, 0.3671512108539464, -1.7116444418612087e-2, -
 0.16255935052964152, 8.284828145987133e-2, 0.25037259184042276, -
 0.23197072459753604, 4.12814896407688e-2, -6.0268019963488524e-
 2, 2.5277070032343814e-2], [-0.28084443831181827, 4.4689094471263374e-2, -
 0.14205911906483717, 0.1674859031551699, 0.28118293671202005, null, -
 2.6948112432816352e-2, 0.43831070877707085, -3.0588050120877303e-
 2, 0.19668173148938084, 9.469730538960319e-
 2, 0.2694611878331789, 0.3258512261810043, -
 0.3338556521279561, 0.4155429641288157, -0.3184852025622286, -
 0.2979749231099391, 0.2005131624140415, 0.2275870179883259, -
 0.24458911735204816, -
 0.31774847979621834, 0.2621769549031754, 0.38685368502889445],
 [2.1515754022741935e-2, -2.2436171272371075e-2, 9.112392165944976e-
 2, 0.15356562992058065, 0.11432632371779464, -2.6948112432816352e-
 2, null, 5.400028577085581e-2, 0.1333800840748123, 8.985073058880731e-
 2, 0.3396726556833403, 0.31872005683752536, 0.3171459202630583, -
 0.11600002544747479, 0.2948873724194819, 0.41195835291927185, -
 0.15861261870440124, 6.668224829883118e-2, 8.972104574107308e-2, -
 0.47180456819736327, 0.12968343497290466, 0.11069612964588414, -
 6.664731810333592e-2], [-0.18645096600808575, -5.506763147227038e-2, -
 4.852039914068734e-
 3, 0.3112017580295911, 0.32783014348784584, 0.43831070877707085, 5.4000285
 77085581e-2, null, 9.120291748558179e-

2, 0.2147826459820511, 0.10395767411608421, 0.34597841574973504, 0.2643159
 132341458, -0.16489254457226926, 0.3798623675225332, -
 0.11462922906360949, -0.18461987559464366, 4.8265478965795285e-
 2, 0.23484427016446696, -0.25518118210090535, -
 0.10344859681112492, 0.20989202645741903, 0.18787236592176484],
 [0.14367627838279415, -0.24246494951667572, 2.280303547527108e-
 3, 0.1143491915729628, 0.1258806309722324, -3.0588050120877303e-
 2, 0.1333800840748123, 9.120291748558179e-2, null, 5.1488926990843634e-2, -
 6.602827508327583e-2, 0.15000385867682697, 0.13935481818002723, -
 4.834405228997633e-
 2, 0.14986639220184336, 0.11073033107675588, 3.319947041431347e-
 2, 5.006735460369927e-2, 8.828705659862363e-2, -
 0.14402746633782965, 0.17836077894389116, -0.10021823048832659, -
 2.5371299748006976e-2], [-1.2429569091361557e-2, -1.0731168454959012e-
 2, -2.479816749290803e-2, 7.548439434283552e-
 2, 0.2150750024954651, 0.19668173148938084, 8.985073058880731e-
 2, 0.2147826459820511, 5.1488926990843634e-2, null, 8.567424842405154e-
 2, 0.14090974798575417, 0.1150603766587678, -
 0.11054268216692735, 0.21465892655826818, 2.921415965312325e-2, -
 0.1217724713374298, 4.493792260995665e-2, 0.7740126740314183, -
 0.17957410372772611, -1.7344455521774833e-2, 3.804149145091901e-
 2, 9.104370860509448e-2], [-0.3071766804640504, 0.11071139941549857, -
 6.13814926769179e-2, 2.4138230288449414e-2, -4.673410496145254e-
 2, 9.469730538960319e-2, 0.3396726556833403, 0.10395767411608421, -
 6.602827508327583e-2, 8.567424842405154e-
 2, null, 0.2911867630542863, 0.16571880837214328, -2.672393071096375e-
 2, 0.14894741824988686, 1.8541012784517287e-2, -
 0.12529804716206575, 1.1836563726706894e-2, 6.33419741088781e-2, -
 0.24782825317584628, -
 0.33987242897632525, 0.3763502089221938, 0.17951531691135422], [-
 0.10045931987772906, -3.505062765102709e-2, -6.8032022252419e-
 2, 0.3700914093188439, 0.2832034213062188, 0.2694611878331789, 0.318720056
 83752536, 0.34597841574973504, 0.15000385867682697, 0.14090974798575417, 0
 .2911867630542863, null, 0.4621599608146992, -
 0.28145939065517483, 0.4848410081222408, 1.8567133146489738e-4, -
 0.26281214071173065, 0.13582507856038015, 0.15804664336297344, -
 0.4247152395146208, -4.493122643967174e-
 3, 0.17885158317541083, 0.14862520966382797], [-
 0.11876775074832747, 3.956082925739585e-3, -
 0.1138859339718653, 0.27881526060368567, 0.2303081142515371, 0.3258512261
 810043, 0.3171459202630583, 0.2643159132341458, 0.13935481818002723, 0.115
 0603766587678, 0.16571880837214328, 0.4621599608146992, null, -
 0.29416233049319584, 0.7772500262288801, 1.1632004786808698e-2, -
 0.24389002415310956, 0.18852934220840542, 0.1396686818791329, -
 0.3645469470905519, 2.6446838468868663e-
 2, 0.14794476065667006, 0.11627950398673828],
 [0.15155170179490057, 5.2340310851350755e-2, 0.5201343920288073, -
 0.22042579963008363, -0.20397424701090383, -0.3338556521279561, -
 0.11600002544747479, -0.16489254457226926, -4.834405228997633e-2, -

```

0.11054268216692735, -2.672393071096375e-2, -0.28145939065517483, -
0.29416233049319584, null, -
0.34852439976065447, 0.29071069540307765, 0.7157502773685008, -
0.6273291273736434, -
0.13857879024373698, 0.19372710728749032, 0.12137121774734226, -
3.668516202002718e-2, -0.2522452637189178], [-0.130858540912874, -
2.0791058855023643e-2, -
0.11879376428796899, 0.3902451915423873, 0.3671512108539464, 0.4155429641
288157, 0.2948873724194819, 0.3798623675225332, 0.14986639220184336, 0.214
65892655826818, 0.14894741824988686, 0.4848410081222408, 0.77725002622888
01, -0.34852439976065447, null, -4.429929863883994e-2, -
0.2826503426365461, 0.20033891571304427, 0.22887216630136714, -
0.3856955827883639, -1.3753032866653308e-
2, 0.14448059014855366, 0.14534461961199352], [0.2538711026953098, -
5.441375162084245e-2, 0.18847556367509205, -4.560653079591508e-2, -
1.7116444418612087e-2, -0.3184852025622286, 0.41195835291927185, -
0.11462922906360949, 0.11073033107675588, 2.921415965312325e-
2, 1.8541012784517287e-2, 1.8567133146489738e-4, 1.1632004786808698e-
2, 0.29071069540307765, -4.429929863883994e-2, null, 0.29972553995326634, -
7.978275433363163e-2, 1.2514447488288668e-2, -
0.10213632891901143, 0.28575940098831115, -0.10930373740673313, -
0.2630050694314128], [0.17978735434939427, -1.5608130072435064e-
2, 0.1528367448304904, -0.1732394296111281, -0.16255935052964152, -
0.2979749231099391, -0.15861261870440124, -
0.18461987559464366, 3.319947041431347e-2, -0.1217724713374298, -
0.12529804716206575, -0.26281214071173065, -
0.24389002415310956, 0.7157502773685008, -
0.2826503426365461, 0.29972553995326634, null, -0.19251374112077824, -
0.1526253538338803, 0.2357430042719733, 0.15199948039902192, -
0.14146241084670816, -0.21459046832418371], [-6.785579219915185e-2, -
8.099300725319886e-2, -0.7193731203252456, 3.534490943430985e-
2, 8.284828145987133e-2, 0.2005131624140415, 6.668224829883118e-
2, 4.8265478965795285e-2, 5.006735460369927e-2, 4.493792260995665e-
2, 1.1836563726706894e-2, 0.13582507856038015, 0.18852934220840542, -
0.6273291273736434, 0.20033891571304427, -7.978275433363163e-2, -
0.19251374112077824, null, 6.642196616448673e-2, -6.556938565561579e-2, -
0.12760018322979433, -8.710460743850316e-2, 0.24045319665624817], [
1.1136665141270262e-2, -4.593800458569556e-4, -2.7079224179623336e-
2, 8.147978558553504e-
2, 0.25037259184042276, 0.2275870179883259, 8.972104574107308e-
2, 0.23484427016446696, 8.828705659862363e-
2, 0.7740126740314183, 6.33419741088781e-
2, 0.15804664336297344, 0.1396686818791329, -
0.13857879024373698, 0.22887216630136714, 1.2514447488288668e-2, -
0.1526253538338803, 6.642196616448673e-2, null, -0.20332810604708884, -
1.5931038643751422e-2, 3.260999258283064e-2, 0.1246195331019028], [-
2.673347148426576e-2, 2.1485586714586398e-2, -3.198067863083505e-2, -
0.22430368544369425, -0.23197072459753604, -0.24458911735204816, -
0.47180456819736327, -0.25518118210090535, -0.14402746633782965, -

```

```

0.17957410372772611, -0.24782825317584628, -0.4247152395146208, -
0.3645469470905519, 0.19372710728749032, -0.3856955827883639, -
0.10213632891901143, 0.2357430042719733, -6.556938565561579e-2, -
0.20332810604708884, null, -7.018005336300569e-2, -0.13195186187743704, -
7.204799150822044e-2], [0.2705620863637573, -
0.14482406000075418, 0.15196903802474168, 2.6144262364923634e-
2, 4.12814896407688e-2, -0.31774847979621834, 0.12968343497290466, -
0.10344859681112492, 0.17836077894389116, -1.7344455521774833e-2, -
0.33987242897632525, -4.493122643967174e-3, 2.6446838468868663e-
2, 0.12137121774734226, -1.3753032866653308e-
2, 0.28575940098831115, 0.15199948039902192, -0.12760018322979433, -
1.5931038643751422e-2, -7.018005336300569e-2, null, -4.7217590828186495e-
2, -0.172166372125269], [-0.6547384167787116, 0.1902256295897464, -
4.88457448222347e-2, 1.2878725817232569e-2, -6.0268019963488524e-
2, 0.2621769549031754, 0.11069612964588414, 0.20989202645741903, -
0.10021823048832659, 3.804149145091901e-
2, 0.3763502089221938, 0.17885158317541083, 0.14794476065667006, -
3.668516202002718e-2, 0.14448059014855366, -0.10930373740673313, -
0.14146241084670816, -8.710460743850316e-2, 3.260999258283064e-2, -
0.13195186187743704, -4.7217590828186495e-2, null, 0.21327318148570945], ,
[-0.29062096521357356, 9.177707540627393e-2, -
0.18196969216223963, 2.9657852023864163e-2, 2.5277070032343814e-
2, 0.38685368502889445, -6.664731810333592e-2, 0.18787236592176484, -
2.5371299748006976e-2, 9.104370860509448e-
2, 0.17951531691135422, 0.14862520966382797, 0.11627950398673828, -
0.2522452637189178, 0.14534461961199352, -0.2630050694314128, -
0.21459046832418371, 0.24045319665624817, 0.1246195331019028, -
7.204799150822044e-2, -
0.172166372125269, 0.21327318148570945, null]]}], "layout": {
    "autosize": true,
    "coloraxis": {"cmax": 1, "cmin": -1, "colorbar": {"title": {"text": "Corrélation"}}, "colorscale": [[0, "rgb(103,0,31)"], [0.1, "rgb(178,24,43)"], [0.2, "rgb(214,96,77)"], [0.3, "rgb(244,165,130)"], [0.4, "rgb(253,219,199)"], [0.5, "rgb(247,247,247)"], [0.6, "rgb(209,229,240)"], [0.7, "rgb(146,197,222)"], [0.8, "rgb(67,147,195)"], [0.9, "rgb(33,102,172)"]], [1, "rgb(5,48,97)"]]}, "height": 800, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, "contour": {"outlineWidth": 0, "ticks": "", "type": "choropleth"}]
}

```

```

[{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "contour"}, {"contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}]}, {"heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmap"}]}, {"heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "heatmapgl"}]}, {"histogram": [{"marker": {"pattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}]}, {"histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2d"}]}, {"histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "histogram2dcontour"}]}, {"mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}]}, {"parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}]}, {"pie": [{"automargin": true, "type": "pie"}]}, {"scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, {"type": "scatter"}]}, {"scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}]}, {"scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}]}, {"scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}]}, {"scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}]}, {"scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}]}

```

```

[{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolar"}}, {"scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolargl"}}, {"scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterternary"}}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FEBC52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, {"scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white"}}

```

```

    "lor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}  

    , "zaxis":  

    {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}}, "title": {"text": "Matrice de corrélation", "x": 0.5}, "width": 800, "xaxis": {"anchor": "y", "constrain": "domain", "domain": [0, 1], "scaleanchor": "y", "tickangle": 45, "title": {"text": "Caractéristiques"}}, "yaxis": {"anchor": "x", "autorange": "reversed", "constrain": "domain", "domain": [0, 1], "title": {"text": "Caractéristiques"}}}}

```

Cette matrice de corrélation révèle des relations entre plusieurs caractéristiques. Notamment :

I2M2 : Corrélé négativement à Cond25, NO2, PO4 et Ptot, indiquant une dégradation biologique lorsque ces paramètres augmentent. Positivement corrélé à l'oxygène dissous et son taux de saturation, soulignant leur importance pour la qualité biologique.

Clustering des stations

Démarche

Dans cette partie, nous cherchons à répondre à notre problématique initiale :

Est-il possible de retrouver les hydroécorégions à partir des données physicochimiques et hydrobiologiques ?

Pour cela, nous avons réalisé un clustering pour regrouper les stations en fonction de leurs caractéristiques. Cette approche nous permet d'identifier des regroupements naturels basés sur les données, susceptibles de refléter les hydroécorégions, tout en capturant des relations complexes et non linéaires que les corrélations simples ne révèlent pas.

Nous avons commencé par un nettoyage des données, incluant la gestion des valeurs manquantes et des variables non numériques. Les données ont été normalisées, et nous avons supprimé les colonnes inutiles ou susceptibles d'influencer artificiellement le clustering, comme les identifiants de stations.

Ensuite, nous avons déterminé le nombre optimal de clusters en utilisant deux méthodes : la méthode du coude (basée sur la diminution de l'inertie) et le coefficient Davies-Bouldin (évaluant la séparation et la compacité des clusters).

Une fois le nombre de clusters défini, nous avons appliqué l'algorithme K-means pour attribuer un cluster à chaque station.

Enfin, nous avons analysé les résultats en comparant les clusters obtenus avec les hydroécorégions afin d'évaluer leur cohérence.

Préparation des données

Gestion des données manquantes

```
df_pc_median_bio_median_1_month_with_coords.isnull().sum()
```

```
station 0
année 0
saison 0
Ammonium - Eau 2109
Azote Kjeldahl - Eau 2512
Carbone Organique - Eau 1748
Conductivité à 25°C - Eau 351
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 2015
Diuron - Eau 9367
Matières en suspension - Eau 371
Nitrates - Eau 2022
Nitrites - Eau 2019
Orthophosphates (P04) - Eau 1957
Oxygène dissous - Eau 407
Phosphore total - Eau 1633
Potentiel en Hydrogène (pH) - Eau 329
Taux de saturation en oxygène - Eau 1351
Température de l'Eau - Eau 284
Turbidité Formazine Néphélométrique - Eau 7048
I2M2 0
CoordXStationMesureEauxSurface 0
CoordYStationMesureEauxSurface 0
dtype: int64
```

```
missing_percentage =
df_pc_median_bio_median_1_month_with_coords.isnull().mean() * 100
print(missing_percentage)
```

```
station 0.000000
année 0.000000
saison 0.000000
Ammonium - Eau 9.557257
Azote Kjeldahl - Eau 11.383514
Carbone Organique - Eau 7.921330
Conductivité à 25°C - Eau 1.590610
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 9.131282
Diuron - Eau 42.447999
Matières en suspension - Eau 1.681243
Nitrates - Eau 9.163004
```

```

Nitrites - Eau                                9.149409
Orthophosphates (P04) - Eau                  8.868446
Oxygène dissous - Eau                         1.844383
Phosphore total - Eau                        7.400190
Potentiel en Hydrogène (pH) - Eau            1.490914
Taux de saturation en oxygène - Eau          6.122264
Température de l'Eau - Eau                   1.286990
Turbidité Formazine Néphélométrique - Eau    31.939095
I2M2                                         0.000000
CoordXStationMesureEauxSurface             0.000000
CoordYStationMesureEauxSurface              0.000000
dtype: float64

# plus de 40 % de données manquantes pour Diuron - Eau
# on peut supprimer cette colonne
df_pc_median_bio_median_1_month_with_coords.drop(columns=['Diuron - Eau'], inplace=True)

# Calculer le nombre de valeurs manquantes par ligne
missing_values_per_row =
df_pc_median_bio_median_1_month_with_coords.isnull().sum(axis=1)
missing_summary = missing_values_per_row.value_counts().reset_index()
missing_summary.columns = ['Nombre de valeurs manquantes', 'Nombre de lignes']
print(missing_summary.sort_values('Nombre de valeurs manquantes'))

      Nombre de valeurs manquantes  Nombre de lignes
0                           0           11776
1                           1            7137
3                           2             823
5                           3             197
9                           4              97
4                           5             261
8                           6             127
7                           7             144
2                           8            1087
12                          9              42
6                           10            193
11                          11              81
15                          12               1
13                          13               8
10                          14              87
14                          15               6

```

Nous avons imputé les données manquantes par leur médiane. Nous avons également testé d'imputer ces données par leur moyenne mais les résultats de clusterings étaient légèrement moins bons.

```
# Imputation des valeurs manquantes avec la médiane
imputer = SimpleImputer(strategy='median') # ici on pourrait aussi
```

```

mettre mean
colonnes = df_pc_median_bio_median_1_month_with_coords.columns
colonnes = colonnes.drop('station')
colonnes = colonnes.drop('année')
colonnes = colonnes.drop('saison')
colonnes = colonnes.drop('I2M2')
df_pc_median_bio_median_1_month_with_coords[colonnes] =
imputer.fit_transform(df_pc_median_bio_median_1_month_with_coords[colonnes])
df_pc_median_bio_median_1_month_with_coords.isnull().sum()

station 0
année 0
saison 0
Ammonium - Eau 0
Azote Kjeldahl - Eau 0
Carbone Organique - Eau 0
Conductivité à 25°C - Eau 0
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 0
Matières en suspension - Eau 0
Nitrates - Eau 0
Nitrites - Eau 0
Orthophosphates (P04) - Eau 0
Oxygène dissous - Eau 0
Phosphore total - Eau 0
Potentiel en Hydrogène (pH) - Eau 0
Taux de saturation en oxygène - Eau 0
Température de l'Eau - Eau 0
Turbidité Formazine Néphélométrique - Eau 0
I2M2 0
CoordXStationMesureEauxSurface 0
CoordYStationMesureEauxSurface 0
dtype: int64

```

Gestion des données non numériques

```

# Mapper chaque saison à un trimestre
saison_to_quarter = {
    "Hiver": 1,
    "Printemps": 2,
    "Été": 3,
    "Automne": 4
}
df_pc_median_bio_median_1_month_with_coords['trimestre'] =
df_pc_median_bio_median_1_month_with_coords['saison'].map(saison_to_quarter)
# drop saison
df_pc_median_bio_median_1_month_with_coords.drop(columns=['saison'],
inplace=True)
df_pc_median_bio_median_1_month_with_coords.head(5)

```

station	année	Ammonium - Eau	Azote Kjeldahl - Eau	\
0	5001800	2007	0.04	1.0
1	5001800	2008	0.04	1.0
2	5001800	2009	0.02	1.0
3	5005350	2007	0.04	1.0
4	5005350	2008	0.03	1.0
		Carbone Organique - Eau	Conductivité à 25°C - Eau	\
0		3.5	765.0	
1		2.7	765.0	
2		2.2	736.0	
3		1.6	600.0	
4		1.6	610.0	
		Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	\	
0			0.5	
1			0.6	
2			1.0	
3			1.9	
4			1.0	
		Matières en suspension - Eau	Nitrates - Eau	Nitrites -
Eau	...	\		
0		3.0	36.2	0.09 ...
1		4.0	40.3	0.07 ...
2		6.0	37.9	0.07 ...
3		22.0	41.7	0.07 ...
4		40.0	40.5	0.07 ...
		Oxygène dissous - Eau	Phosphore total - Eau	\
0		8.1	0.05	
1		8.3	0.11	
2		7.4	0.07	
3		7.3	0.33	
4		8.9	0.25	
		Potentiel en Hydrogène (pH) - Eau	Taux de saturation en oxygène -	
Eau	\			
0			7.9	
89.0				
1			7.9	
90.0				
2			8.0	
83.0				
3			8.1	
76.0				

```

4                               8.1
86.0

    Température de l'Eau - Eau   Turbidité Formazine Néphélométrique -
Eau \
0                      18.6
4.3
1                      20.5
4.3
2                      20.6
4.3
3                      18.6
4.3
4                      14.2
4.3

    I2M2  CoordXStationMesureEauxSurface
CoordYStationMesureEauxSurface \
0  0.3004                  399856.0
6531980.0
1  0.3848                  399856.0
6531980.0
2  0.5756                  399856.0
6531980.0
3  0.4851                  450151.0
6572920.0
4  0.3488                  450151.0
6572920.0

    trimestre
0      3
1      3
2      3
3      3
4      3

[5 rows x 21 columns]

# Convertir l'identifiant de station en variable catégorielle (cela crée un encodage numérique)
df_pc_median_bio_median_1_month_with_coords['station'] =
df_pc_median_bio_median_1_month_with_coords['station'].astype(str)
df_pc_median_bio_median_1_month_with_coords['trimestre'] =
df_pc_median_bio_median_1_month_with_coords['trimestre'].astype(int)
df_pc_median_bio_median_1_month_with_coords['année'] =
df_pc_median_bio_median_1_month_with_coords['année'].astype(int)
df_pc_median_bio_median_1_month_with_coords.dtypes

station                                object
année                                 int64

```

Ammonium - Eau	float64
Azote Kjeldahl - Eau	float64
Carbone Organique - Eau	float64
Conductivité à 25°C - Eau	float64
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	float64
Matières en suspension - Eau	float64
Nitrates - Eau	float64
Nitrites - Eau	float64
Orthophosphates (P04) - Eau	float64
Oxygène dissous - Eau	float64
Phosphore total - Eau	float64
Potentiel en Hydrogène (pH) - Eau	float64
Taux de saturation en oxygène - Eau	float64
Température de l'Eau - Eau	float64
Turbidité Formazine Néphéломétrique - Eau	float64
I2M2	float64
CoordXStationMesureEauxSurface	float64
CoordYStationMesureEauxSurface	float64
trimestre	int64
dtype: object	

Suppression des colonnes

Nous supprimons les colonnes des coordonnées, afin de ne pas influencer le clustering.

```
df_pc_median_bio_median_1_month_c =
df_pc_median_bio_median_1_month_with_coords.copy()

df_pc_median_bio_median_1_month_c =
df_pc_median_bio_median_1_month_with_coords.drop(columns=[ 'CoordXStationMesureEauxSurface' , 'CoordYStationMesureEauxSurface' ])
df_clustering = df_pc_median_bio_median_1_month_c.copy()
```

Normalisation et encodage des données

Pour préserver l'information relative aux stations, essentielle pour suivre l'évolution de leurs caractéristiques physicochimiques et hydrobiologiques sur plusieurs trimestres, nous avons choisi de conserver cette colonne dans le processus de clustering.

Pour minimiser tout biais lié aux identifiants de stations, nous avons encodé cette colonne avec des valeurs numériques arbitraires. Ensuite, comme pour les autres caractéristiques, nous avons normalisé ses valeurs afin d'assurer une contribution équilibrée au clustering.

Nous avons également testé la suppression de la colonne station, mais cela a légèrement dégradé les résultats. Par ailleurs, perdre le lien temporel qu'offre cette information entre les enregistrements nous semblait non pertinent. Nous avons donc décidé de conserver la colonne station, en sachant qu'elle introduit un léger biais.

```
df_clustering.head(5)
```

station	année	Ammonium - Eau	Azote Kjeldahl - Eau	\
0	5001800	2007	0.04	1.0
1	5001800	2008	0.04	1.0
2	5001800	2009	0.02	1.0
3	5005350	2007	0.04	1.0
4	5005350	2008	0.03	1.0
		Carbone Organique - Eau	Conductivité à 25°C - Eau	\
0		3.5	765.0	
1		2.7	765.0	
2		2.2	736.0	
3		1.6	600.0	
4		1.6	610.0	
		Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	\	
0			0.5	
1			0.6	
2			1.0	
3			1.9	
4			1.0	
		Matières en suspension - Eau	Nitrates - Eau	Nitrites - Eau \
0		3.0	36.2	0.09
1		4.0	40.3	0.07
2		6.0	37.9	0.07
3		22.0	41.7	0.07
4		40.0	40.5	0.07
		Orthophosphates (P04) - Eau	Oxygène dissous - Eau	Phosphore total - Eau \
0		0.05		8.1
0.05				
1		0.05		8.3
0.11				
2		0.05		7.4
0.07				
3		0.54		7.3
0.33				
4		0.27		8.9
0.25				
		Potentiel en Hydrogène (pH) - Eau	Taux de saturation en oxygène - Eau \	
0			7.9	
89.0				
1			7.9	
90.0				
2			8.0	
83.0				
3			8.1	

```

76.0
4                               8.1
86.0

    Température de l'Eau - Eau   Turbidité Formazine Néphéломétrique -
Eau \
0                      18.6
4.3
1                      20.5
4.3
2                      20.6
4.3
3                      18.6
4.3
4                      14.2
4.3

    I2M2   trimestre
0  0.3004            3
1  0.3848            3
2  0.5756            3
3  0.4851            3
4  0.3488            3

```

Nous utilisons LabelEncoder plutôt que One-Hot Encoding en raison du volume important de données. One-Hot Encoding aurait entraîné une explosion du nombre de colonnes, ce qui aurait considérablement alourdi le traitement et augmenté la complexité du modèle. LabelEncoder permet de représenter les catégories sous forme d'entiers, ce qui est plus efficace en termes de mémoire et de calcul.

Étant donné que nous travaillons avec des données dont la distribution est bornée et sans connaître précisément leur forme, nous utilisons le MinMaxScaler. Ce dernier est plus adapté pour redimensionner les données dans une plage spécifique lorsque celles-ci sont déjà bornées.

```

# Encodage des stations
le = LabelEncoder()
df_clustering['station_encoded'] =
le.fit_transform(df_clustering['station'])
df_clustering.drop(columns=['station'], inplace=True)
# Normalisation des données pour qu'elles aient toutes la même
importance
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(df_clustering)
normalized_data = pd.DataFrame(normalized_data,
columns=df_clustering.columns)

normalized_data.head(5)

    année  Ammonium - Eau   Azote Kjeldahl - Eau   Carbone Organique -
Eau \

```

0	0.000000	0.006169	0.190283
0.220588			
1	0.066667	0.006169	0.190283
0.167112			
2	0.133333	0.002742	0.190283
0.133690			
3	0.000000	0.006169	0.190283
0.093583			
4	0.066667	0.004455	0.190283
0.093583			

Conductivité à 25°C - Eau \

0	0.423398
1	0.423398
2	0.407242
3	0.331476
4	0.337047

Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \

0	0.000000
1	0.007407
2	0.037037
3	0.103704
4	0.037037

Matières en suspension - Eau Nitrates - Eau Nitrites - Eau \

0	0.005657	0.614274	0.096045
1	0.007833	0.683942	0.073446
2	0.012185	0.643161	0.073446
3	0.046997	0.707732	0.073446
4	0.086162	0.687341	0.073446

Orthophosphates (P04) - Eau Oxygène dissous - Eau Phosphore total - Eau \

0	0.018265	0.456140
0.032258		
1	0.018265	0.467836
0.075269		
2	0.018265	0.415205
0.046595		
3	0.242009	0.409357
0.232975		
4	0.118721	0.502924
0.175627		

Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène - Eau \

0	0.516854
0.453039	
1	0.516854

```

0.464088
2 0.539326
0.386740
3 0.561798
0.309392
4 0.561798
0.419890

    Température de l'Eau - Eau Turbidité Formazine Néphélométrique -
Eau \
0 0.588571
0.012003
1 0.642857
0.012003
2 0.645714
0.012003
3 0.588571
0.012003
4 0.462857
0.012003

      I2M2 trimestre station_encoded
0 0.3004 0.666667 0.380424
1 0.3848 0.666667 0.380424
2 0.5756 0.666667 0.380424
3 0.4851 0.666667 0.380741
4 0.3488 0.666667 0.380741

```

Recherche du nombre de clusters optimal

```

nb_hydroecoregions = df_hydroregions['CdHER1'].nunique()
print("Nombre d'hydroécorégions : ", nb_hydroecoregions)

Nombre d'hydroécorégions :  22

# Plage des valeurs de k à tester
k_values = range(nb_hydroecoregions-5, nb_hydroecoregions+5)

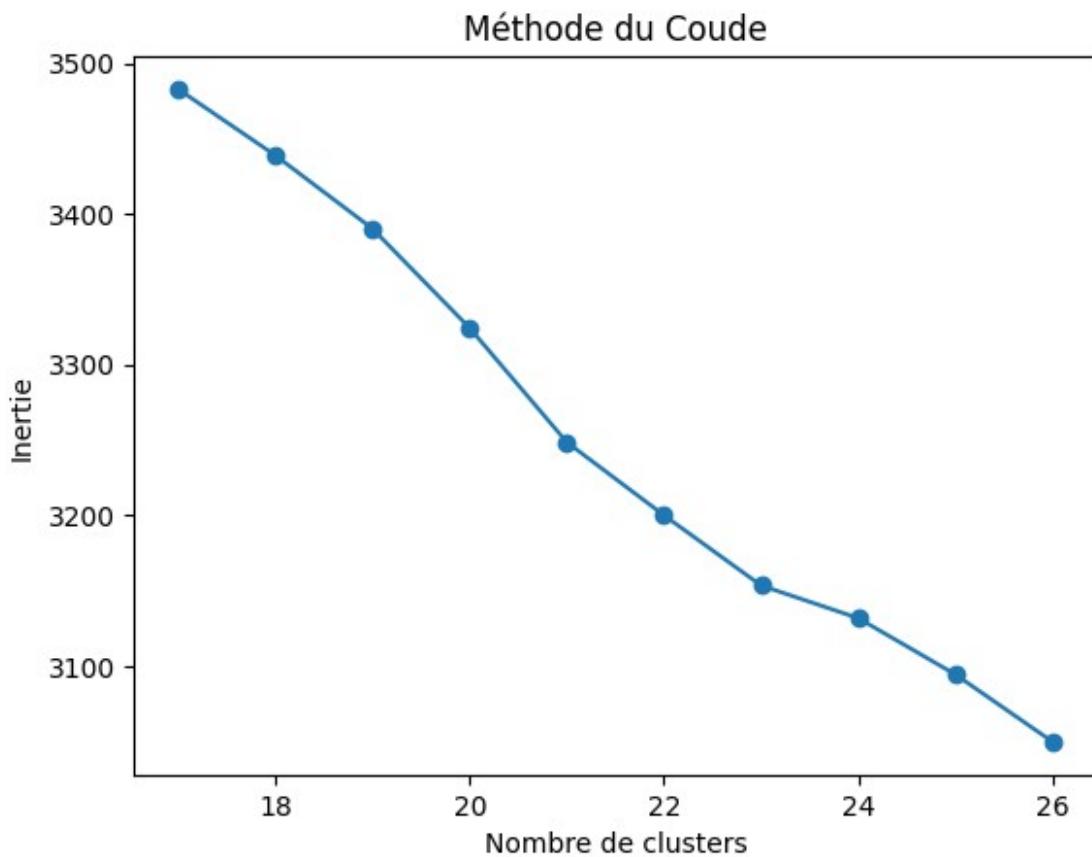
# Calcule de l'inertie et du score de Davies-Bouldin pour chaque
# valeur de k
davies_bouldin_scores = []
inertia = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(normalized_data)
    inertia.append(kmeans.inertia_)
    cluster_labels = kmeans.fit_predict(normalized_data)
    davies_bouldin_scores.append(davies_bouldin_score(normalized_data,
cluster_labels))

```

Méthode du coude

La méthode du coude consiste à observer la diminution de l'inertie (somme des erreurs quadratiques intra-clusters) en fonction du nombre de clusters. Le point où la diminution devient moins significative peut indiquer le nombre optimal de clusters.

```
# Tracer l'inertie pour observer le coude
plt.plot(k_values, inertia, marker='o')
plt.title('Méthode du Coude')
plt.xlabel('Nombre de clusters')
plt.ylabel('Inertie')
plt.show()
```



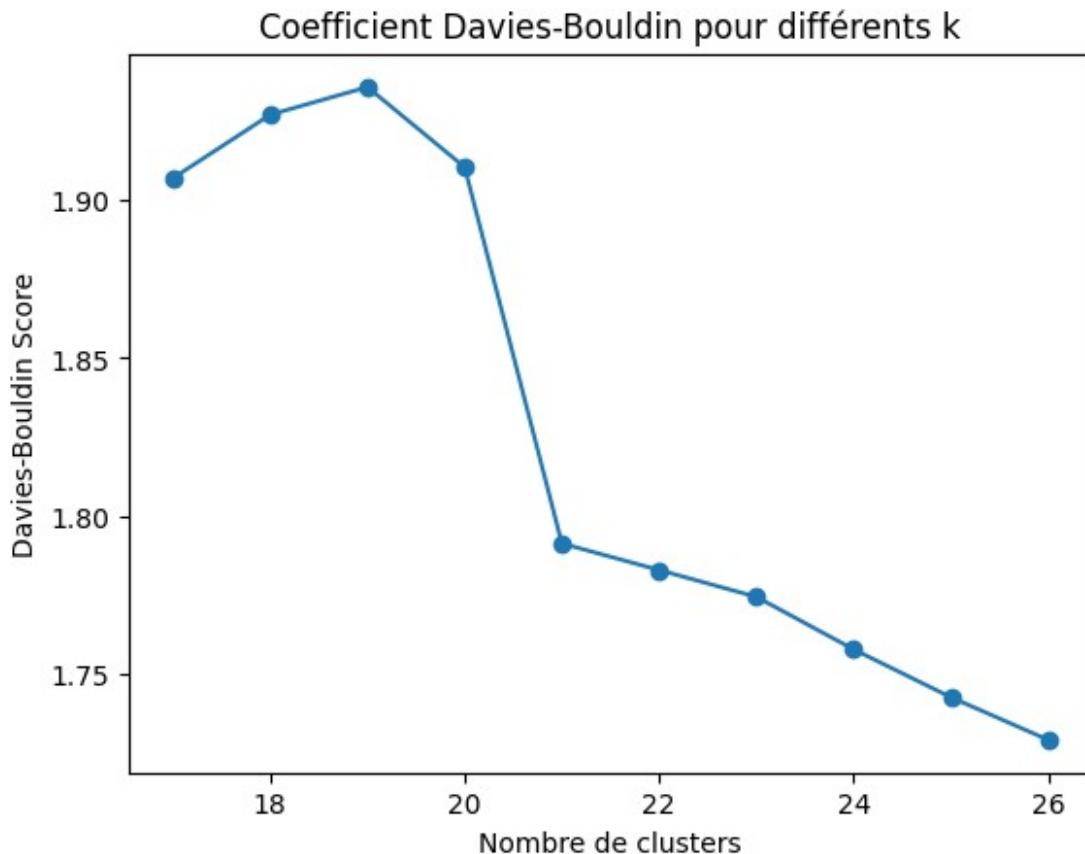
Nous n'arrivons pas à voir de coude, cela rend impossible à déterminer le nombre optimal de clusters. Nous allons donc utiliser le coefficient Davies-Bouldin pour faire notre choix.

Coefficient Davies-Bouldin

Le coefficient Davies-Bouldin évalue la qualité des clusters en mesurant leur compacité et leur séparation. Une valeur plus faible indique des clusters mieux définis.

```
plt.plot(k_values, davies_bouldin_scores, marker='o')
plt.title('Coefficient Davies-Bouldin pour différents k')
```

```
plt.xlabel('Nombre de clusters')
plt.ylabel('Davies-Bouldin Score')
plt.show()
```



Nous pouvons voir clairement que le score de Davies-Bouldin est minimal pour 26 clusters (pour k testé entre 2 et 40).

On voit tout de même une nette amélioration autour de 21. Nous allons donc utiliser k = 22 pour notre clustering, car le but est de retrouver les hydroécorégions.

Réalisation du clustering avec K-Means

Clustering

```
nb_clusters = 22 # c'est le nombre d'hydroecoregions
kmeans = KMeans(n_clusters=nb_clusters, random_state=42)
kmeans.fit(normalized_data)
KMeans(n_clusters=22, random_state=42)
```

Analyse des caractéristiques des clusters

```
# Analyse des centres des clusters
cluster_centers = pd.DataFrame(kmeans.cluster_centers_,
columns=df_clustering.columns)
print("Centres des clusters :")
print(cluster_centers)
# Variabilité des caractéristiques par cluster
influence = cluster_centers.std(axis=0)
print("\nInfluence des caractéristiques :")
print(influence.sort_values(ascending=False))

Centres des clusters :
    année Ammonium - Eau Azote Kjeldahl - Eau Carbone Organique
- Eau \
0   0.707556      0.003651      0.102529
0.090147
1   0.249482      0.017077      0.160705
0.176372
2   0.262536      0.014060      0.175244
0.183344
3   0.762536      0.006252      0.109104
0.174547
4   0.209027      0.011974      0.199577
0.091468
5   0.733449      0.003293      0.106691
0.040919
6   0.748257      0.008057      0.104491
0.146719
7   0.638630      0.006876      0.107028
0.188756
8   0.665855      0.014854      0.129444
0.157090
9   0.247924      0.009221      0.191398
0.090527
10  0.717722      0.027798      0.133328
0.198282
11  0.807137      0.006971      0.114634
0.204066
12  0.526005      0.005426      0.115513
0.098832
13  0.652447      0.018494      0.206541
0.481690
14  0.762981      0.009198      0.118056
0.178326
15  0.192699      0.009375      0.172560
0.134697
16  0.894674      0.006302      0.107940
0.158099
17  0.556588      0.007803      0.109166
```

0.120521		
18 0.700532	0.021140	0.157779
0.245159		
19 0.826806	0.004704	0.101173
0.076801		
20 0.276536	0.009791	0.191287
0.082039		
21 0.355509	0.018202	0.176868
0.122463		
22 0.814249	0.015142	0.120401
0.155828		
23 0.260950	0.023928	0.205889
0.205656		
24 0.868313	0.012506	0.134622
0.246087		
25 0.480108	0.152589	0.290769
0.257444		

Conductivité à 25°C - Eau \

0	0.083917
1	0.273867
2	0.351371
3	0.120180
4	0.213629
5	0.205073
6	0.295166
7	0.064199
8	0.266297
9	0.277629
10	0.403002
11	0.241226
12	0.142874
13	0.154753
14	0.315303
15	0.108710
16	0.110466
17	0.196895
18	0.346247
19	0.269281
20	0.207648
21	0.345499
22	0.366028
23	0.269317
24	0.224428
25	0.465191

Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \

0	0.065462
1	0.148145

2	0.084765
3	0.082664
4	0.066397
5	0.038033
6	0.082900
7	0.077674
8	0.099575
9	0.050025
10	0.118933
11	0.074474
12	0.064749
13	0.133186
14	0.074629
15	0.091896
16	0.072650
17	0.071597
18	0.095554
19	0.060225
20	0.049710
21	0.069180
22	0.085742
23	0.115146
24	0.092289
25	0.192835

	Matières en suspension - Eau	Nitrates - Eau	Nitrites - Eau	\
0	0.007899	0.044622	0.009600	
1	0.030106	0.185566	0.051271	
2	0.038301	0.572979	0.074315	
3	0.022764	0.154591	0.026569	
4	0.048794	0.076050	0.036805	
5	0.051281	0.043485	0.011608	
6	0.034108	0.439895	0.049346	
7	0.016506	0.109936	0.017648	
8	0.023499	0.169269	0.075657	
9	0.019808	0.073187	0.028329	
10	0.041505	0.439202	0.126813	
11	0.015615	0.233315	0.042379	
12	0.014491	0.077725	0.016417	
13	0.048744	0.329355	0.075760	
14	0.030557	0.629380	0.070677	
15	0.015747	0.125854	0.027065	
16	0.025116	0.155928	0.019269	
17	0.019049	0.159901	0.023966	
18	0.031388	0.214918	0.108319	
19	0.013003	0.082618	0.015652	
20	0.018757	0.063177	0.022403	
21	0.025858	0.169976	0.056839	
22	0.025760	0.299379	0.084097	

23	0 . 052828	0 . 221993	0 . 067103
24	0 . 040994	0 . 181510	0 . 048739
25	0 . 042953	0 . 284341	0 . 304347
Orthophosphates (P04) - Eau Oxygène dissous - Eau Phosphore			
total - Eau \			
0	0 . 017006	0 . 676854	
0 . 009912			
1	0 . 058611	0 . 632617	
0 . 073694			
2	0 . 052002	0 . 618264	
0 . 048502			
3	0 . 029990	0 . 613244	
0 . 027831			
4	0 . 024063	0 . 593761	
0 . 025254			
5	0 . 021155	0 . 575365	
0 . 023051			
6	0 . 059522	0 . 640152	
0 . 043455			
7	0 . 031890	0 . 694866	
0 . 025926			
8	0 . 066942	0 . 602821	
0 . 054702			
9	0 . 031468	0 . 574686	
0 . 025530			
10	0 . 112139	0 . 621864	
0 . 083887			
11	0 . 057067	0 . 494650	
0 . 043200			
12	0 . 017520	0 . 619107	
0 . 015612			
13	0 . 068471	0 . 617529	
0 . 073423			
14	0 . 045624	0 . 619389	
0 . 040201			
15	0 . 028789	0 . 661249	
0 . 024234			
16	0 . 026509	0 . 654008	
0 . 025550			
17	0 . 029588	0 . 657775	
0 . 025109			
18	0 . 162713	0 . 477894	
0 . 116622			
19	0 . 024304	0 . 669451	
0 . 013972			
20	0 . 023531	0 . 673789	
0 . 021888			
21	0 . 068901	0 . 644093	

0.047917		
22	0.088712	0.633728
0.058378		
23	0.066508	0.652213
0.061345		
24	0.049033	0.606033
0.046663		
25	0.391780	0.528359
0.293074		

Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène - Eau \		
0	0.529470	
0.578352		
1	0.597472	
0.556958		
2	0.634363	
0.520178		
3	0.533527	
0.576700		
4	0.635238	
0.576303		
5	0.662548	
0.590869		
6	0.621018	
0.538580		
7	0.457653	
0.569557		
8	0.627375	
0.569984		
9	0.628274	
0.518665		
10	0.619666	
0.504149		
11	0.529128	
0.420317		
12	0.610069	
0.582813		
13	0.471684	
0.500551		
14	0.598479	
0.527340		
15	0.531853	
0.552623		
16	0.500066	
0.558306		
17	0.612487	
0.560486		
18	0.585888	

0.430193	
19	0.673847
0.588631	
20	0.631848
0.569483	
21	0.635705
0.549654	
22	0.641091
0.532795	
23	0.616638
0.533716	
24	0.549007
0.487083	
25	0.615253
0.425290	

Température de l'Eau - Eau Turbidité Formazine Néphéломétrique - Eau \	
0	0.324424
0.008366	
1	0.341541
0.023065	
2	0.416521
0.016675	
3	0.493606
0.019382	
4	0.531091
0.014535	
5	0.515841
0.045804	
6	0.393688
0.029119	
7	0.282177
0.016420	
8	0.522290
0.024770	
9	0.485407
0.014219	
10	0.370021
0.037037	
11	0.546561
0.015890	
12	0.457672
0.014606	
13	0.398580
0.051454	
14	0.435577
0.029328	
15	0.327763

0.012605	
16	0.362893
0.022876	
17	0.353490
0.016402	
18	0.590467
0.035210	
19	0.371030
0.014813	
20	0.344982
0.015967	
21	0.392227
0.018649	
22	0.393392
0.028040	
23	0.347125
0.017469	
24	0.403048
0.042699	
25	0.434076
0.026171	

	I2M2	trimestre	station_encoded
0	0.788561	3.030303e-02	0.833181
1	0.482370	6.215722e-02	0.060374
2	0.492051	5.698006e-02	0.521086
3	0.744246	3.358950e-01	0.333032
4	0.590951	4.380952e-01	0.709719
5	0.659711	7.828107e-01	0.747919
6	0.660685	5.646788e-02	0.144228
7	0.734467	2.471751e-03	0.192480
8	0.396767	3.327536e-01	0.690459
9	0.606577	9.988138e-01	0.746061
10	0.240064	4.135021e-02	0.109760
11	0.517291	9.992407e-01	0.302635
12	0.774137	3.361793e-01	0.617687
13	0.528996	2.300293e-02	0.356155
14	0.558662	5.714286e-02	0.424124
15	0.718474	9.553054e-03	0.508475
16	0.717312	2.775558e-17	0.408255
17	0.657733	-4.163336e-16	0.538017
18	0.292821	9.946752e-01	0.760685
19	0.613654	2.159345e-02	0.814597
20	0.652630	2.570033e-03	0.826063
21	0.309488	1.403509e-02	0.862052
22	0.253129	3.816794e-03	0.770339
23	0.304609	1.681379e-02	0.570321
24	0.304851	2.652035e-02	0.335984
25	0.220997	4.704301e-02	0.742990

```
Influence des caractéristiques :
trimestre                                0.340712
station_encoded                            0.247609
année                                      0.236529
I2M2                                       0.184560
Nitrates - Eau                             0.157295
Conductivité à 25°C - Eau                  0.103313
Carbone Organique - Eau                     0.085707
Température de l'Eau - Eau                  0.079395
Orthophosphates (P04) - Eau                 0.074581
Nitrites - Eau                               0.059011
Potentiel en Hydrogène (pH) - Eau          0.058338
Phosphore total - Eau                      0.055093
Oxygène dissous - Eau                      0.053010
Taux de saturation en oxygène - Eau        0.049228
Azote Kjeldahl - Eau                       0.046795
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 0.033240
Ammonium - Eau                             0.028363
Matières en suspension - Eau                0.013082
Turbidité Formazine Néphéломétrique - Eau 0.011083
dtype: float64
```

```
# Affichage des caractéristiques avec les plus grandes variabilités
entre clusters
# = Caractéristiques les plus influentes
fig = px.bar(influence.sort_values(ascending=False),
              title="Influence des caractéristiques par cluster",
              labels={'index': 'Caractéristiques', 'value': "Écart-type
des centres des clusters"})
fig.update_layout(xaxis_tickangle=45, width=800, height=600,
showlegend=False)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data":
[{"alignmentgroup": "True", "hovertemplate": "variable=0<br>Caractéristiques=%{x}<br>Écart-type des centres des clusters=%{y}<extra></extra>", "legendgroup": "0", "marker": {"color": "#636efa", "pattern": {"shape": ""}}, "name": "0", "offsetgroup": "0", "orientation": "v", "showlegend": true, "textposition": "auto", "type": "bar", "x": ["trimestre", "station_encoded", "année", "I2M2", "Nitrates - Eau", "Conductivité à 25°C - Eau", "Carbone Organique - Eau", "Température de l'Eau - Eau", "Orthophosphates (P04) - Eau", "Nitrites - Eau", "Potentiel en Hydrogène (pH) - Eau", "Phosphore total - Eau", "Oxygène dissous - Eau", "Taux de saturation en oxygène - Eau", "Azote Kjeldahl - Eau", "Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau", "Ammonium - Eau", "Matières en suspension - Eau", "Turbidité Formazine Néphéломétrique - Eau"], "xaxis": "x", "y": [0.3407121672463686, 0.24760914974373274, 0.23652863464522755, 0.18456020]
```

```

409788812, 0.15729469596623993, 0.1033133016307297, 8.570686677021819e-
2, 7.939530682552697e-2, 7.458109532209965e-2, 5.901106325353187e-
2, 5.833764996065531e-2, 5.5093118982329665e-2, 5.3010373215782884e-
2, 4.922760656895299e-2, 4.679458416962482e-2, 3.324049246511823e-
2, 2.8363389442420596e-2, 1.3081655074428778e-2, 1.1082921951301297e-
2], "yaxis": "y"}], "layout": {"barmode": "relative", "height": 600, "legend": {"title": {"text": "variable"}, "tracegroupgap": 0}, "showlegend": false, "template": {"data": [{"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "bar"}}, {"barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "barpolar"}]}, {"carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}, {"choropleth": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "choropleth"}]}, {"contour": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}], {"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "contourcarpet"}]}, {"heatmap": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}], {"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "heatmapgl": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}], {"colorbar": {"outlinewidth": 0, "ticks": ""}, "type": "histogram": [{"marker": {"pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "histogram"}}, {"histogram2d": [{"colorbar": {"outlinewidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}]}]}]}]
```

```

[1, "#f0f921"]], "type": "histogram2d"}, "histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}, [1, "#f0f921"]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"automargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}, "type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}, [1, "#f0f921"]], "type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}, "type": "table"}}, {"layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autoTypeNumbers": "strict", "colorAxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]]}, "sequential": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"]]}]}]
```

```
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus":  
[[0, "#0d0887"], [0.1111111111111111, "#46039f"],  
[0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"],  
[0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"],  
[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],  
[0.8888888888888888, "#fdca26"], [1, "#f0f921"]]], "colorway":  
["#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692",  
"#B6E880", "#FF97FF", "#FECB52"], "font": {"color": "#2a3f5f"}, "geo":  
{"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true,  
"showland": true, "subunitcolor": "white"}, "hoverlabel":  
{"align": "left"}, "hovermode": "closest", "mapbox":  
{"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis":  
{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis":  
{"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene":  
{"xaxis":  
{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  
"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis":  
{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  
"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis":  
{"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white",  
"showbackground": true, "ticks": "", "zerolinecolor": "white"}, "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis":  
{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis":  
{"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "caxis":  
{"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title":  
{"x": 5.0e-2}, "xaxis":  
{"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  
{"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis":  
{"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  
{"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}, "title":  
{"text": "Influence des caractéristiques par  
cluster"}, "width": 800, "xaxis": {"anchor": "y", "domain": [0, 1], "tickangle": 45, "title": {"text": "Caractéristiques"}}, "yaxis":  
{"anchor": "x", "domain": [0, 1], "title": {"text": "Écart-type des centres  
des clusters"}}}}
```

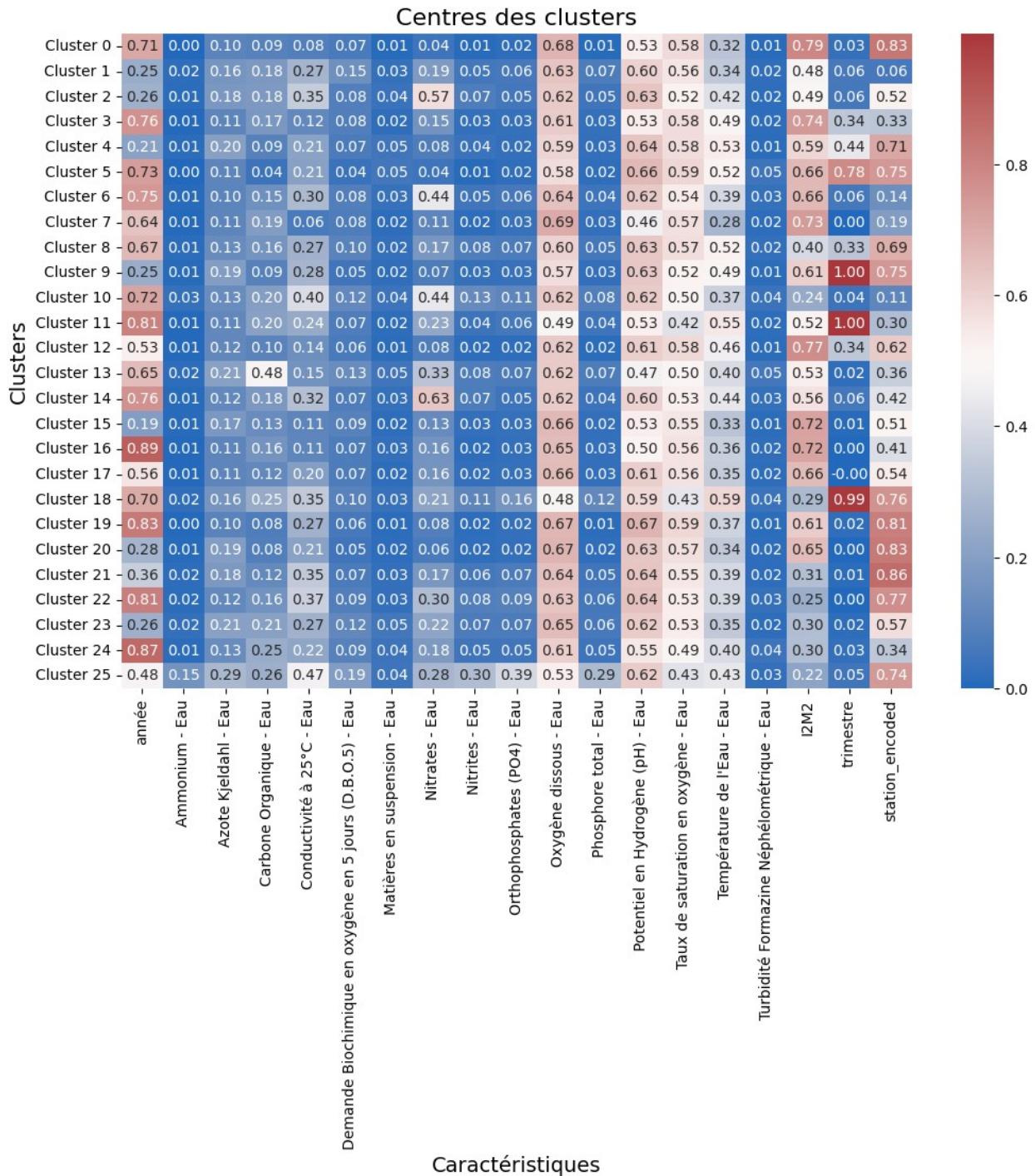
En dehors des données spatio-temporelles, les paramètres ayant le plus d'impact sont l'indice I2M2, les Nitrates, la Température de l'eau et la Conductivité à 25°C.

```
# Matrice des centres des clusters  
plt.figure(figsize=(12, 8))  
sns.heatmap(cluster_centers, annot=True, fmt=".2f", cmap="vlag",
```

```

xticklabels=df_clustering.columns, yticklabels=[f'Cluster {i}' for i
in range(len(cluster_centers))])
plt.title("Centres des clusters", fontsize=16)
plt.xlabel("Caractéristiques", fontsize=14)
plt.ylabel("Clusters", fontsize=14)
plt.show()

```



La matrice des centres des clusters révèle des valeurs disparates pour l'indice I2M2, les Nitrates, le Carbone organique et la Conductivité de l'eau. En revanche, pour certaines colonnes, nous obtenons des valeurs identiques, avec une ou deux valeurs dominantes dans l'ensemble des clusters.

Attribution des clusters

```

df_pc_median_bio_median_1_month_with_clusters =
df_pc_median_bio_median_1_month_c.copy()

df_pc_median_bio_median_1_month_with_clusters['cluster'] =
kmeans.labels_

df_pc_median_bio_median_1_month_with_clusters.head(5)

    station   année   Ammonium - Eau   Azote Kjeldahl - Eau \
0  5001800  2007           0.04           1.0
1  5001800  2008           0.04           1.0
2  5001800  2009           0.02           1.0
3  5005350  2007           0.04           1.0
4  5005350  2008           0.03           1.0

    Carbone Organique - Eau   Conductivité à 25°C - Eau \
0                   3.5           765.0
1                   2.7           765.0
2                   2.2           736.0
3                   1.6           600.0
4                   1.6           610.0

    Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \
0                           0.5
1                           0.6
2                           1.0
3                           1.9
4                           1.0

    Matières en suspension - Eau   Nitrates - Eau   Nitrites - Eau \
0                   3.0           36.2           0.09
1                   4.0           40.3           0.07
2                   6.0           37.9           0.07
3                  22.0           41.7           0.07
4                  40.0           40.5           0.07

    Orthophosphates (P04) - Eau   Oxygène dissous - Eau   Phosphore total \
- Eau \
0                   0.05           8.1
0.05
1                   0.05           8.3
0.11
2                   0.05           7.4
0.07

```

3	0.54	7.3
0.33		
4	0.27	8.9
0.25		
Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène -		
Eau \		
0	7.9	
89.0		
1	7.9	
90.0		
2	8.0	
83.0		
3	8.1	
76.0		
4	8.1	
86.0		
Température de l'Eau - Eau Turbidité Formazine Néphéломétrique -		
Eau \		
0	18.6	
4.3		
1	20.5	
4.3		
2	20.6	
4.3		
3	18.6	
4.3		
4	14.2	
4.3		
I2M2	trimestre	cluster
0	0.3004	3
1	0.3848	3
2	0.5756	3
3	0.4851	3
4	0.3488	3
		9
		9
		9
		9
		9

Calcul du score de silhouette

```
# Calculer le score de la silhouette
score = silhouette_score(normalized_data,
df_pc_median_bio_median_1_month_with_clusters['cluster'])
print(f"Silhouette score: {score}")

Silhouette score: 0.1251427074415718
```

Le score de silhouette obtenu est plutôt faible, ce qui indique que les stations ne sont pas clairement regroupées dans leurs clusters respectifs, et que certaines stations sont plus proches des centres d'autres clusters.

```

# Recherche du meilleur nombre de clusters selon le score de
silhouette
silhouette_scores = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(normalized_data)

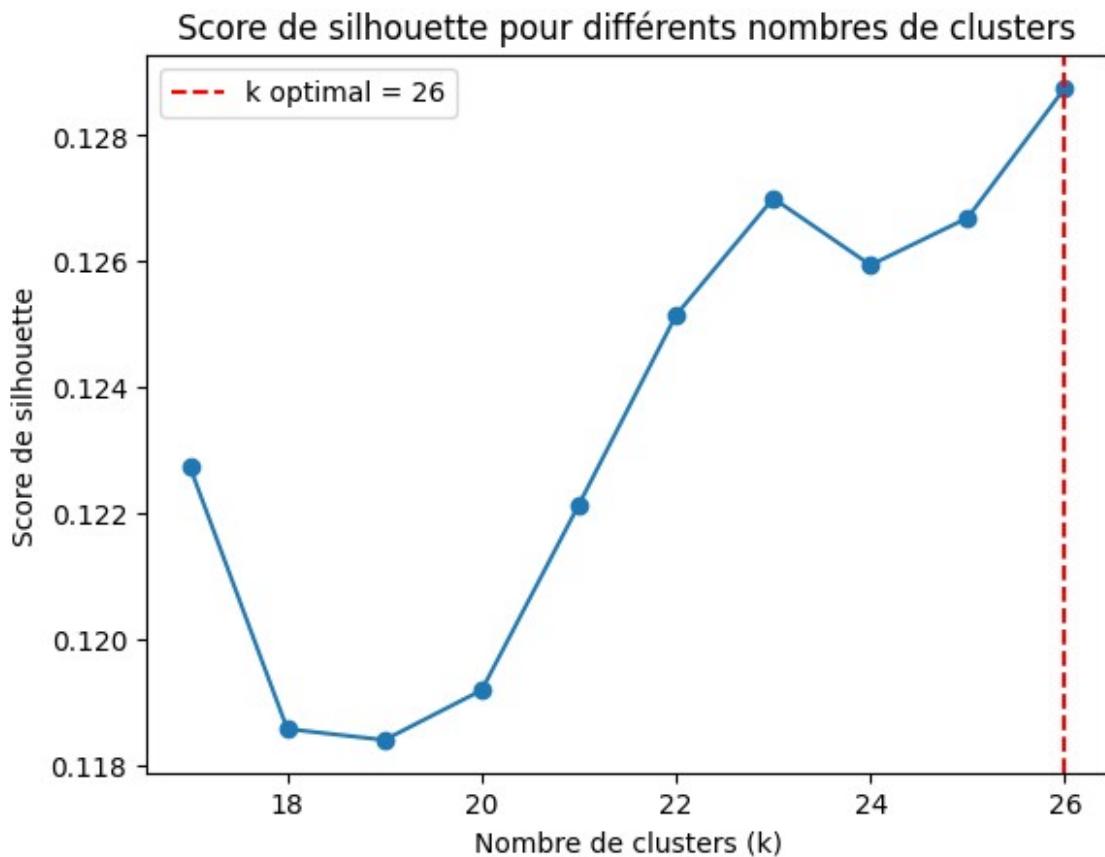
    score = silhouette_score(normalized_data, cluster_labels)
    silhouette_scores.append(score)

best_k = k_values[silhouette_scores.index(max(silhouette_scores))]
print(f"Le meilleur nombre de clusters est : {best_k}")

Le meilleur nombre de clusters est : 26

plt.plot(k_values, silhouette_scores, marker='o')
plt.title("Score de silhouette pour différents nombres de clusters")
plt.xlabel("Nombre de clusters (k)")
plt.ylabel("Score de silhouette")
plt.axvline(x=best_k, color='r', linestyle='--', label=f"k optimal = {best_k}")
plt.legend()
plt.show()

```



Le score de silhouette est le meilleur quand k = 26. Nous conservons k = 22, afin de rester alignée avec notre problématique géographique.

Etude des résultats obtenus

Chaque station a-t-elle été assignée à un seul cluster ?

```
# Créer un DataFrame pour calculer le pourcentage d'appartenance des stations aux clusters
cluster_distribution =
df_pc_median_bio_median_1_month_with_clusters.groupby('station')
['cluster'].value_counts(normalize=True).unstack().fillna(0) * 100
print(cluster_distribution)
```

cluster	0	1	2	3	4	5	6	7	\
station									
1000477	16.666667	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	
1000602	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	
1000605	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	
1001122	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	
1001131	0.000000	0.0	0.000000	0.0	0.0	0.000000	0.0	0.0	
...	
6999125	0.000000	0.0	7.142857	0.0	0.0	28.571429	0.0	0.0	
6999137	0.000000	0.0	20.000000	60.0	0.0	20.000000	0.0	0.0	
6999153	0.000000	0.0	60.000000	0.0	0.0	40.000000	0.0	0.0	
6999176	0.000000	0.0	100.000000	0.0	0.0	0.000000	0.0	0.0	
6999178	0.000000	0.0	100.000000	0.0	0.0	0.000000	0.0	0.0	
cluster	8	9	...	12	13	14	15	16	
17 \									
station	...								
1000477	16.666667	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
66.666667	0.000000	100.000000	0.0	...	0.000000	0.000000	0.0	0.0	
1000602	100.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
0.000000	16.666667	0.0	...	50.000000	0.000000	0.0	0.0	0.0	
0.000000	0.000000	0.0	...	16.666667	0.000000	0.0	0.0	0.0	
1001122	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
0.000000	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
1001131	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
66.666667	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
...	
6999125	0.000000	0.0	...	0.000000	7.142857	0.0	0.0	0.0	
0.000000	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
6999137	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
0.000000	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
6999153	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
0.000000	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	
6999176	0.000000	0.0	...	0.000000	0.000000	0.0	0.0	0.0	

```
0.000000
6999178    0.000000  0.0  ...  0.000000  0.000000  0.0  0.0  0.0
0.000000
```

```
cluster      18        19        20        21
station
1000477    0.000000  0.000000  0.000000  0.0
1000602    0.000000  0.000000  0.000000  0.0
1000605  33.333333  0.000000  0.000000  0.0
1001122    0.000000  83.333333  0.000000  0.0
1001131    0.000000  33.333333  0.000000  0.0
...
6999125    0.000000  0.000000  21.428571  0.0
6999137    0.000000  0.000000  0.000000  0.0
6999153    0.000000  0.000000  0.000000  0.0
6999176    0.000000  0.000000  0.000000  0.0
6999178    0.000000  0.000000  0.000000  0.0
```

```
[3158 rows x 22 columns]
```

```
# combien de lignes dans le dataframe ?
df_pc_median_bio_median_1_month_with_clusters.shape[0]

22067

# combien de stations différentes ?
df_pc_median_bio_median_1_month_with_clusters['station'].nunique()

3158
```

```
stations_100_percent =
cluster_distribution[cluster_distribution.eq(100.0).sum(axis=1) == 1]
print("Stations à 100% dans un seul cluster : ", end=' ')
print(stations_100_percent.shape[0])
print("Pourcentage de stations 100% dans le même cluster : ",
round(stations_100_percent.shape[0] /
df_pc_median_bio_median_1_month_with_clusters['station'].nunique() * 100, 2), "%")
```

```
Stations à 100% dans un seul cluster : 778
Pourcentage de stations 100% dans le même cluster : 24.64 %
```

Nous voyons qu'un quart des stations sont classées dans le même cluster pour tous les trimestres et toutes les années. Bien que cela ne soit pas beaucoup, cela montre que le clustering prend en compte la station et s'en sert pour relier les enregistrement et suivre les variations trimestrielles.

Combien d'enregistrements classés dans la bonne hydrorégion ?

```
# Attribution des clusters au dataset contenant les coordonnées des
stations
```

```

df_pc_median_bio_median_1_month_with_clusters_with_coord =
df_pc_median_bio_median_1_month_with_coords.copy()
df_pc_median_bio_median_1_month_with_clusters_with_coord['cluster'] =
kmeans.labels_

# Attribution du cluster dominant à chaque station
station_cluster_counts =
df_pc_median_bio_median_1_month_with_clusters_with_coord.groupby(['sta
tion', 'cluster']).size().reset_index(name='count')
dominant_cluster_per_station =
station_cluster_counts.loc[station_cluster_counts.groupby('station')
['count'].idxmax()]
df_pc_median_bio_median_1_month_with_clusters_with_coord =
df_pc_median_bio_median_1_month_with_clusters_with_coord.merge(dominan
t_cluster_per_station[['station', 'cluster']], on='station',
how='left', suffixes=('', '_dominant'))
df_pc_median_bio_median_1_month_with_clusters_with_coord.rename(column
s={'cluster_dominant': 'cluster_dominant_station'}, inplace=True)

# Conservation des colonnes nécessaires
df_analyse =
df_pc_median_bio_median_1_month_with_clusters_with_coord[['station',
'année', 'CoordXStationMesureEauxSurface',
'CoordYStationMesureEauxSurface', 'trimestre',
'cluster_dominant_station']].copy()

# Ajout de l'hydroécorégion réelle pour chaque station
carto_i2m2_her = gpd.GeoDataFrame(df_analyse, crs=crs_lambert,
geometry=gpd.GeoSeries(df_analyse.agg(lambda x:
geom.Point(x.loc['CoordXStationMesureEauxSurface'],
x.loc['CoordYStationMesureEauxSurface']), axis=1)))
HER_stations_her =
carto_i2m2_her.sjoin(df_hydroregions.to_crs(crs_lambert),
predicate='within').to_crs(crs_lambert)
df_analyse_etude = HER_stations_her.drop(columns=['geometry',
'index_right', 'gid'])
df_analyse_etude.rename(columns={'CdHER1': 'hydroecoregion'},
inplace=True)

# Pour chaque cluster, nous déterminons quelle est l'hydrorégion dominante
dominant_class_per_cluster =
df_analyse_etude.groupby('cluster_dominant_station')
['hydroecoregion'].agg(lambda x: x.mode()[0]).reset_index()
dominant_class_per_cluster.columns = ['cluster_dominant_station',
'dominant_hydroecoregion_cluster']
df_analyse_etude = df_analyse_etude.merge(dominant_class_per_cluster,
on='cluster_dominant_station', how='left')

```

```

df_analyse_etude.head(5)

   station  année  CoordXStationMesureEauxSurface \
0  5001800  2007                  399856.0
1  5001800  2008                  399856.0
2  5001800  2009                  399856.0
3  5005350  2007                  450151.0
4  5005350  2008                  450151.0

   CoordYStationMesureEauxSurface  trimestre  cluster_dominant_station \
\0                               6531980.0          3                      9
1                               6531980.0          3                      9
2                               6531980.0          3                      9
3                               6572920.0          3                      9
4                               6572920.0          3                      9

   hydroecoregion      NomHER1  dominant_hydroecoregion_cluster \
0            9    TABLES CALCAIRES                         9
1            9    TABLES CALCAIRES                         9
2            9    TABLES CALCAIRES                         9
3            9    TABLES CALCAIRES                         9
4            9    TABLES CALCAIRES                         9

# On garde qu'une ligne par station
df_analyse_etude.drop_duplicates(subset='station', inplace=True)

# Quel cluster est associé à quelle hydroécorégion ?
clusters_hydroregions =
df_analyse_etude.groupby('cluster_dominant_station')
['hydroecoregion'].apply(lambda x: x.mode()[0]).reset_index()
clusters_hydroregions.columns = ['cluster_dominant_station',
'dominant_hydroecoregion_cluster']
clusters_hydroregions['dominant_hydroecoregion_cluster'] =
clusters_hydroregions['dominant_hydroecoregion_cluster'].astype(str)
df_hydroregions['CdHER1'] = df_hydroregions['CdHER1'].astype(str)
clusters_hydroregions_with_names =
clusters_hydroregions.merge(df_hydroregions[['CdHER1', 'NomHER1']],
left_on='dominant_hydroecoregion_cluster', right_on='CdHER1',
how='left')
clusters_hydroregions_with_names =
clusters_hydroregions_with_names.rename(columns={'NomHER1':
'nom_dominant_hydroecoregion_cluster'})
print(clusters_hydroregions_with_names[['cluster_dominant_station',
'dominant_hydroecoregion_cluster',
'nom_dominant_hydroecoregion_cluster']])

```

	cluster_dominant_station	dominant_hydroecoregion_cluster	\
0	0	6	
1	1	6	
2	2	5	
3	3	6	
4	4	12	
5	5	14	
6	6	6	
7	7	5	
8	8	9	
9	9	9	
10	10	5	
11	11	3	
12	12	10	
13	13	6	
14	14	12	
15	15	9	
16	16	10	
17	17	10	
18	18	9	
19	19	9	
20	20	15	
21	21	2	
22	22	2	
23	23	9	
24	24	6	
25	25	12	

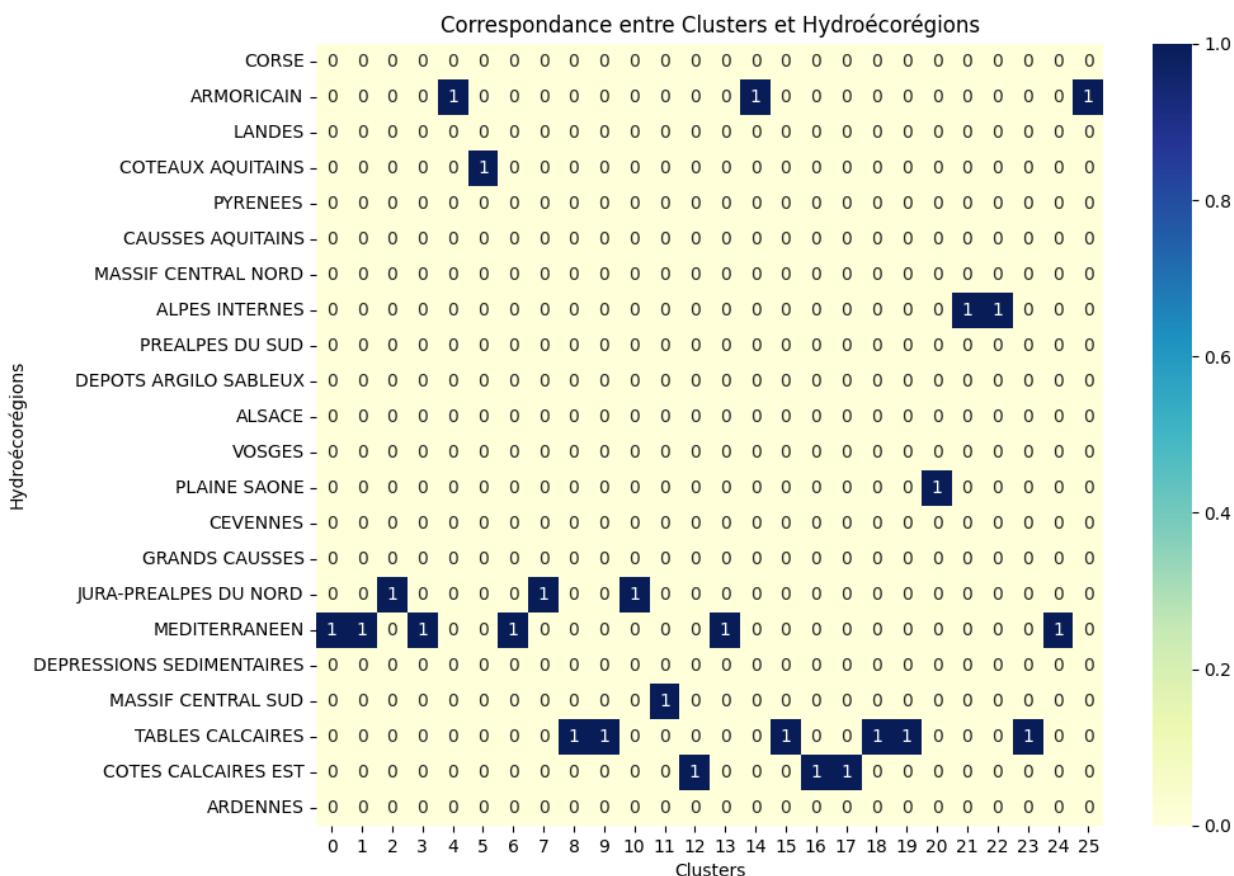
	nom_dominant_hydroecoregion_cluster
0	MEDITERRANEEN
1	MEDITERRANEEN
2	JURA-PREALPES DU NORD
3	MEDITERRANEEN
4	ARMORICAIN
5	COTEAUX AQUITAINS
6	MEDITERRANEEN
7	JURA-PREALPES DU NORD
8	TABLES CALCAIRES
9	TABLES CALCAIRES
10	JURA-PREALPES DU NORD
11	MASSIF CENTRAL SUD
12	COTES CALCAIRES EST
13	MEDITERRANEEN
14	ARMORICAIN
15	TABLES CALCAIRES
16	COTES CALCAIRES EST
17	COTES CALCAIRES EST
18	TABLES CALCAIRES
19	TABLES CALCAIRES
20	PLAINE SAONE

```

21                      ALPES INTERNES
22                      ALPES INTERNES
23                      TABLES CALCAIRES
24                      MEDITERRANEEN
25                      ARMORICAIN

all_hydroecoregions = df_hydroregions['NomHER1'].unique()
cross_tab =
pd.crosstab(clusters_hydroregions_with_names['nom_dominant_hydroecoregion_cluster'],
clusters_hydroregions_with_names['cluster_dominant_station']).reindex(
index=all_hydroecoregions, fill_value=0)
plt.figure(figsize=(10, 8))
sns.heatmap(cross_tab, annot=True, fmt='d', cmap='YlGnBu', cbar=True)
plt.title("Correspondance entre Clusters et Hydroécorégions")
plt.xlabel("Clusters")
plt.ylabel("Hydroécorégions")
plt.show()

```



Nous observons que certaines hydroécorégions sont associées à plusieurs clusters, tandis que d'autres ne sont pas représentées dans les résultats du clustering. Cela peut s'expliquer par des similarités entre les caractéristiques physicochimiques et hydrobiologiques de stations

appartenant à des hydroécorégions différentes, ou par des variations internes importantes au sein d'une même hydroécorégion.

De plus, nous voyons que parmi les 22 clusters définis, seuls 9 des hydroécorégions sont identifiées dans les données.

```
# Ajouter une colonne qui indique si l'enregistrement a bien été classée
# On vérifie pour chaque ligne si le cluster correspond à l'hydroécorégion dominante
df_analyse_etude['correct_classification'] =
df_analyse_etude['hydroecoregion'] ==
df_analyse_etude['dominant_hydroecoregion_cluster']

num_correctly_classified =
df_analyse_etude['correct_classification'].sum()
total_stations = df_analyse_etude.shape[0]
accuracy = num_correctly_classified / total_stations
print(f"Nombre de stations bien classées: {num_correctly_classified}/{total_stations}")
print(f"Taux de classification correcte: {accuracy * 100:.2f}%")

Nombre de stations bien classées: 1091/3139
Taux de classification correcte: 34.76%
```

Quels trimestres présentent le plus de stations bien classées ?

```
df_analyse_etude['annee_trimestre'] =
df_analyse_etude['année'].astype(str) + "-T" +
df_analyse_etude['trimestre'].astype(str)

stations_correct =
df_analyse_etude[df_analyse_etude['correct_classification']]
stations_correct_count = stations_correct.groupby('annee_trimestre')[['station']].nunique().reset_index()
stations_correct_count.columns = ['annee_trimestre',
'stations_correctement_classées']

total_stations_count = df_analyse_etude.groupby('annee_trimestre')[['station']].nunique().reset_index()
total_stations_count.columns = ['annee_trimestre', 'stations_totales']

merged_counts = pd.merge(total_stations_count, stations_correct_count,
on='annee_trimestre', how='left')
merged_counts['stations_correctement_classées'] =
merged_counts['stations_correctement_classées'].fillna(0) # Remplir les valeurs manquantes par 0
merged_counts['pourcentage_correct'] =
(merged_counts['stations_correctement_classées'] /
merged_counts['stations_totales']) * 100
```

```

fig = px.bar(merged_counts, x='annee_trimestre',
y='pourcentage_correct', title='Pourcentage de stations correctement
classées par Année-Trimestre',labels={'annee_trimestre': 'Année -
Trimestre', 'stations_correctement_classées': 'Nombre de Stations
Correctement Classées'},text='stations_correctement_classées')
fig.update_layout(xaxis_tickangle=45, width=900, height=600,
showlegend=False, xaxis_title="Année - Trimestre",
yaxis_title="Pourcentage de stations correctement classées")
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data": [
{"alignmentgroup": "True", "hovertemplate": "Année - Trimestre=%
{x}<br>pourcentage_correct=%{y}<br>Nombre de Stations Correctement
Classées=%{text}<extra></extra>", "legendgroup": "", "marker": {
"color": "#636efa", "pattern": {
"shape": ""}}, "name": "", "offsetgroup": "", "orientation": "v", "showlegend": false, "text": [
76, 1, 22, 27, 172, 15, 2, 9, 22, 1, 2, 14, 0, 4, 20, 0, 6, 8, 11, 1, 5, 38, 1, 3, 12, 2, 22, 12
7, 249, 9, 1, 20, 75, 1, 6, 25, 1, 2, 20, 1, 14, 14, 0, 5, 3, 0, 6, 6, 6, 0, 4], "textposition": "auto", "type": "bar", "x": ["2007-T3", "2007-T4", "2008-T1", "2008-
T2", "2008-T3", "2008-T4", "2009-T1", "2009-T2", "2009-T3", "2010-T1", "2010-
T2", "2010-T3", "2010-T4", "2011-T2", "2011-T3", "2011-T4", "2012-T1", "2012-
T2", "2012-T3", "2012-T4", "2013-T2", "2013-T3", "2014-T1", "2014-T2", "2014-
T3", "2014-T4", "2015-T1", "2015-T2", "2015-T3", "2015-T4", "2016-T1", "2016-
T2", "2016-T3", "2016-T4", "2017-T2", "2017-T3", "2017-T4", "2018-T2", "2018-
T3", "2018-T4", "2019-T2", "2019-T3", "2019-T4", "2020-T2", "2020-T3", "2020-
T4", "2021-T1", "2021-T2", "2021-T3", "2022-T2", "2022-
T3"], "xaxis": "x", "y": [
55.072463768115945, 50, 56.41025641025641, 49.09090909090909, 28.05872756
933116, 28.30188679245283, 100, 52.94117647058824, 30.136986301369863, 100,
25, 33.33333333333333, 0, 44.44444444444444, 40, 0, 66.66666666666666, 38.095
238095238095, 9.734513274336283, 33.33333333333333, 19.230769230769234, 27
.536231884057973, 100, 20, 24.489795918367346, 40, 62.857142857142854, 38.25
301204819277, 45.3551912568306, 36, 100, 30.76923076923077, 29.411764705882
355, 20, 37.5, 26.595744680851062, 100, 12.5, 28.169014084507044, 10, 73.68421
052631578, 22.22222222222222, 0, 45.45454545454545, 14.285714285714285, 0, 1
00, 22.22222222222222, 27.27272727272727, 0, 66.66666666666666], "yaxis": "y
"}], "layout": {"barmode": "relative", "height": 600, "legend": {
"tracegroupgap": 0}, "showlegend": false, "template": {"data": {"bar": [
{"error_x": {"color": "#2a3f5f"}, "error_y": {
"color": "#2a3f5f"}, "marker": {"line": {
"color": "#E5ECF6", "width": 0.5}, "pattern": {
"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "bar"}], "barpolar": [
{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {
"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [
{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min
orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {
"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "min
orgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}]}}, "ch

```

```

oropleth": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "contour"}], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "heatmapgl"}], "histogram": [{"marker": {"pattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}], "histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "histogram2d"}], "histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], "type": "histogram2dcontour"}], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}], "parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"automargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}, "type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}]

```

```

[{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scattergl"}, "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scattermapbox"}}, {"scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolar"}}, {"scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterpolar"}}, {"scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "scatterternary"}}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]], {"type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, "type": "table"}}], "layout": {"annotationDefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], {"sequential": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}, {"sequentialminus": [[[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], [1, "#f0f921"]]}], "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECB52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlakes": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, {"scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}

```

```

    , "yaxis":  

    {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}  

    , "zaxis":  

    {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}  

    , "shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis":  

    {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis":  

    {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "caxis":  

    {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "title":  

    {"x": 5.0e-2}, "xaxis":  

    {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  

    {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}, "yaxis":  

    {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title":  

    {"standoff": 15}, "zerolinecolor": "white", "zerolinewidth": 2}}, "title":  

    {"text": "Pourcentage de stations correctement classées par Année-Trimestre"}, "width": 900, "xaxis": {"anchor": "y", "domain": [0, 1], "tickangle": 45, "title": {"text": "Année - Trimestre"}}, "yaxis": {"anchor": "x", "domain": [0, 1], "title": {"text": "Pourcentage de stations correctement classées"}}}}

```

Visualisation du clustering sur les stations

```

# Représentation des stations avec la couleur de leur cluster dominant  

crs_lambert =  

'PROJCS["RGF_1993_Lambert_93",GEOGCS["GCS_RGF_1993",DATUM["D_RGF_1993",  

SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0]  

,UNIT["Degree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],  

PARAMETER["False_Easting",700000.0],PARAMETER["False_Northing",660  

0000.0],PARAMETER["Central_Meridian",3.0],PARAMETER["Standard_Parallel_1",49.0],  

PARAMETER["Standard_Parallel_2",44.0],PARAMETER["Latitude_of_Origin",46.5],UNIT["Meter",1.0]]'  

x_col = 'CoordXStationMesureEauxSurface'  

y_col = 'CoordYStationMesureEauxSurface'  

cluster_col = 'cluster_dominant_station'  

carto_i2m2 = gpd.GeoDataFrame(df_analyse_etude, crs=crs_lambert,  

geometry=gpd.GeoSeries(df_analyse_etude.apply(lambda x:  

geom.Point(x[x_col], x[y_col]), axis=1)))  

HER_lambert = df_hydroregions.to_crs(crs_lambert)  

unique_clusters = carto_i2m2[cluster_col].unique()  

unique_clusters = sorted(unique_clusters)  

cmap = cm.get_cmap('tab20', len(unique_clusters))  

cluster_colors = {cluster: mcolors.to_hex(cmap(i)) for i, cluster in  

enumerate(unique_clusters)}  

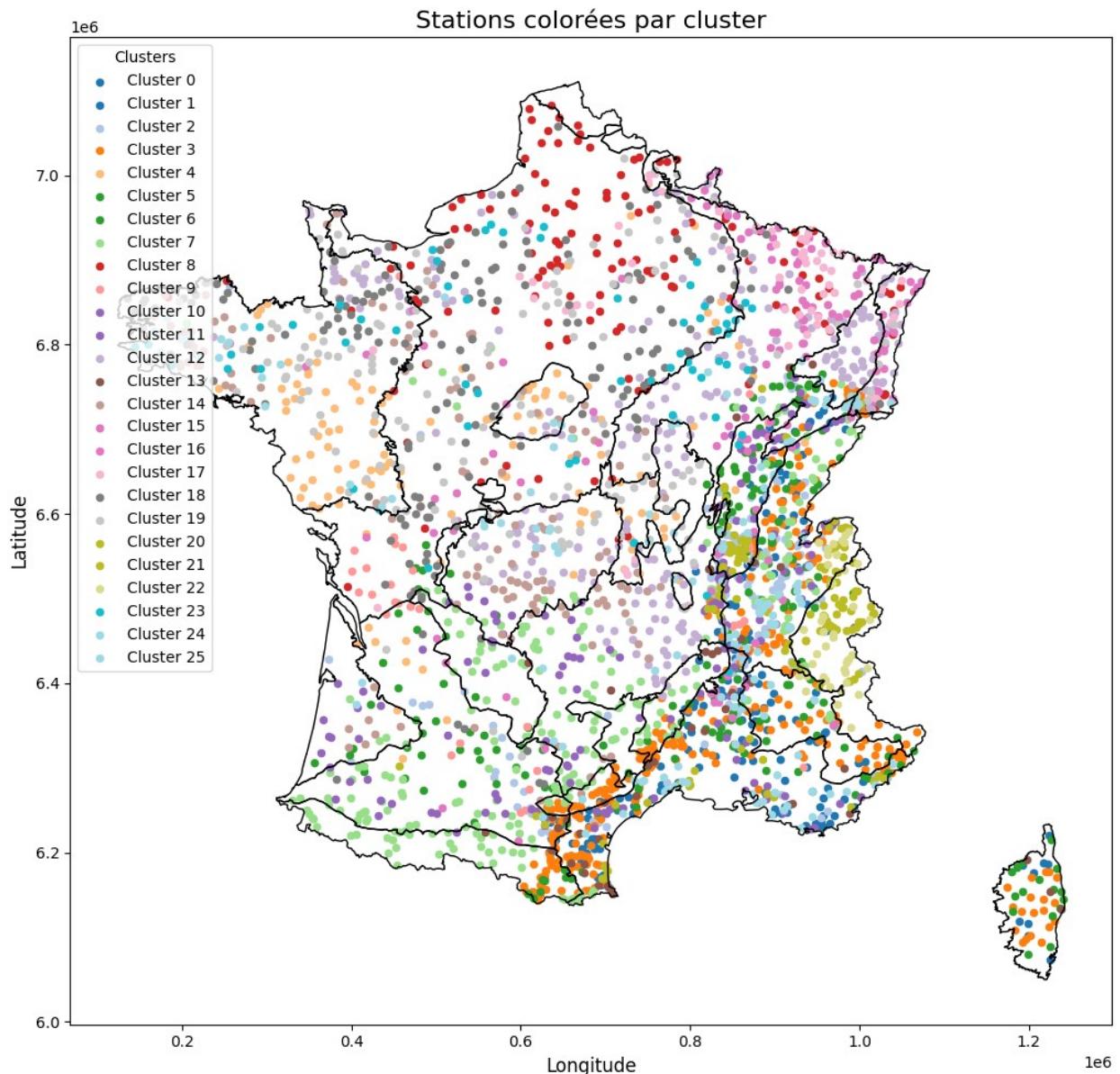
fig, ax = plt.subplots(1, 1, figsize=(12, 10))  

HER_lambert.boundary.plot(ax=ax, color='black', linewidth=1)

```

```
for cluster, color in cluster_colors.items():
    cluster_data = carto_i2m2[carto_i2m2[cluster_col] == cluster]
    cluster_data.plot(ax=ax, color=color, markersize=20,
label=f"Cluster {cluster}")
ax.legend(loc='upper left', fontsize='medium', title='Clusters')
plt.title('Stations colorées par cluster', fontsize=16)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.tight_layout()
plt.show()

/var/folders/p1/4yqk4cyx3058m3j04rtkr56m0000gn/T/
ipykernel_53919/1842718937.py:10: MatplotlibDeprecationWarning:
The get_cmap function was deprecated in Matplotlib 3.7 and will be
removed in 3.11. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```



```
df_analyse_etude.head(5)
```

	station	année	CoordXStationMesureEauxSurface	\
0	5001800	2007	399856.0	
3	5005350	2007	450151.0	
17	5005400	2007	446087.0	
31	5005950	2007	451708.0	
45	5006100	2007	459757.0	

	cluster_dominant_station	CoordYStationMesureEauxSurface	trimestre
0		6531980.0	3
9			

```

3           6572920.0      3
9
17          6568450.0      3
18
31          6565540.0      3
9
45          6561330.0      3
9

    hydroecoregion      NomHER1  dominant_hydroecoregion_cluster
\0            9  TABLES CALCAIRES
9
3            9  TABLES CALCAIRES
9
17           9  TABLES CALCAIRES
9
31           9  TABLES CALCAIRES
9
45           9  TABLES CALCAIRES
9

    correct_classification annee_trimestre
0             True  2007-T3
3             True  2007-T3
17            True  2007-T3
31            True  2007-T3
45            True  2007-T3

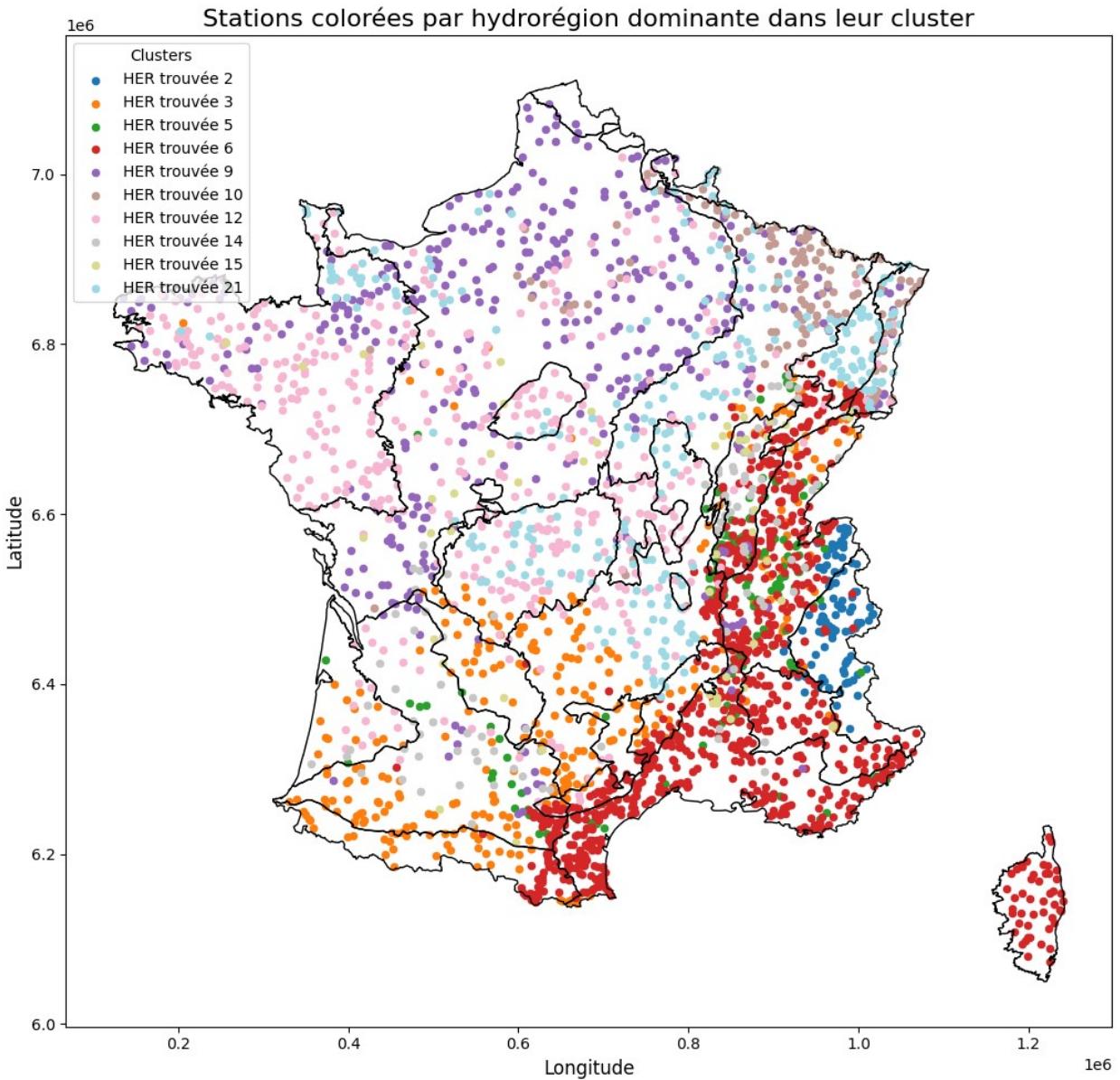
# Représentation des stations avec la couleur de l'hydrorégion dominante dans le cluster
crs_lambert =
'PROJCS["RGF_1993_Lambert_93",GEOGCS["GCS_RGF_1993",DATUM["D_RGF_1993",SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Lambert_Conformal_Conic"],PARAMETER["False_Easting",700000.0],PARAMETER["False_Northing",660000.0],PARAMETER["Central_Meridian",3.0],PARAMETER["Standard_Parallel_1",49.0],PARAMETER["Standard_Parallel_2",44.0],PARAMETER["Latitude_of_Origin",46.5],UNIT["Meter",1.0]]'
x_col = 'CoordXStationMesureEauxSurface'
y_col = 'CoordYStationMesureEauxSurface'
cluster_col = 'dominant_hydroecoregion_cluster'
carto_i2m2 = gpd.GeoDataFrame(df_analyse_etude, crs=crs_lambert, geometry=gpd.GeoSeries(df_analyse_etude.apply(lambda x: geom.Point(x[x_col], x[y_col]), axis=1)))
HER_lambert = df_hydroregions.to_crs(crs_lambert)
unique_clusters = carto_i2m2[cluster_col].unique()
unique_clusters = sorted(unique_clusters)
cmap = cm.get_cmap('tab20', len(unique_clusters))
cluster_colors = {cluster: mcolors.to_hex(cmap(i)) for i, cluster in

```

```
enumerate(unique_clusters)}
fig, ax = plt.subplots(1, 1, figsize=(12, 10))
HER_lambert.boundary.plot(ax=ax, color='black', linewidth=1)
for cluster, color in cluster_colors.items():
    cluster_data = carto_i2m2[carto_i2m2[cluster_col] == cluster]
    cluster_data.plot(ax=ax, color=color, markersize=20, label=f"HER trouvée {cluster}")
ax.legend(loc='upper left', fontsize='medium', title='Clusters')
plt.title('Stations colorées par hydrorégion dominante dans leur cluster', fontsize=16)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.tight_layout()
plt.show()
```

```
/var/folders/p1/4yqk4cyx3058m3j04rtkr56m0000gn/T/
ipykernel_53919/811036617.py:10: MatplotlibDeprecationWarning:
```

```
The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.
```



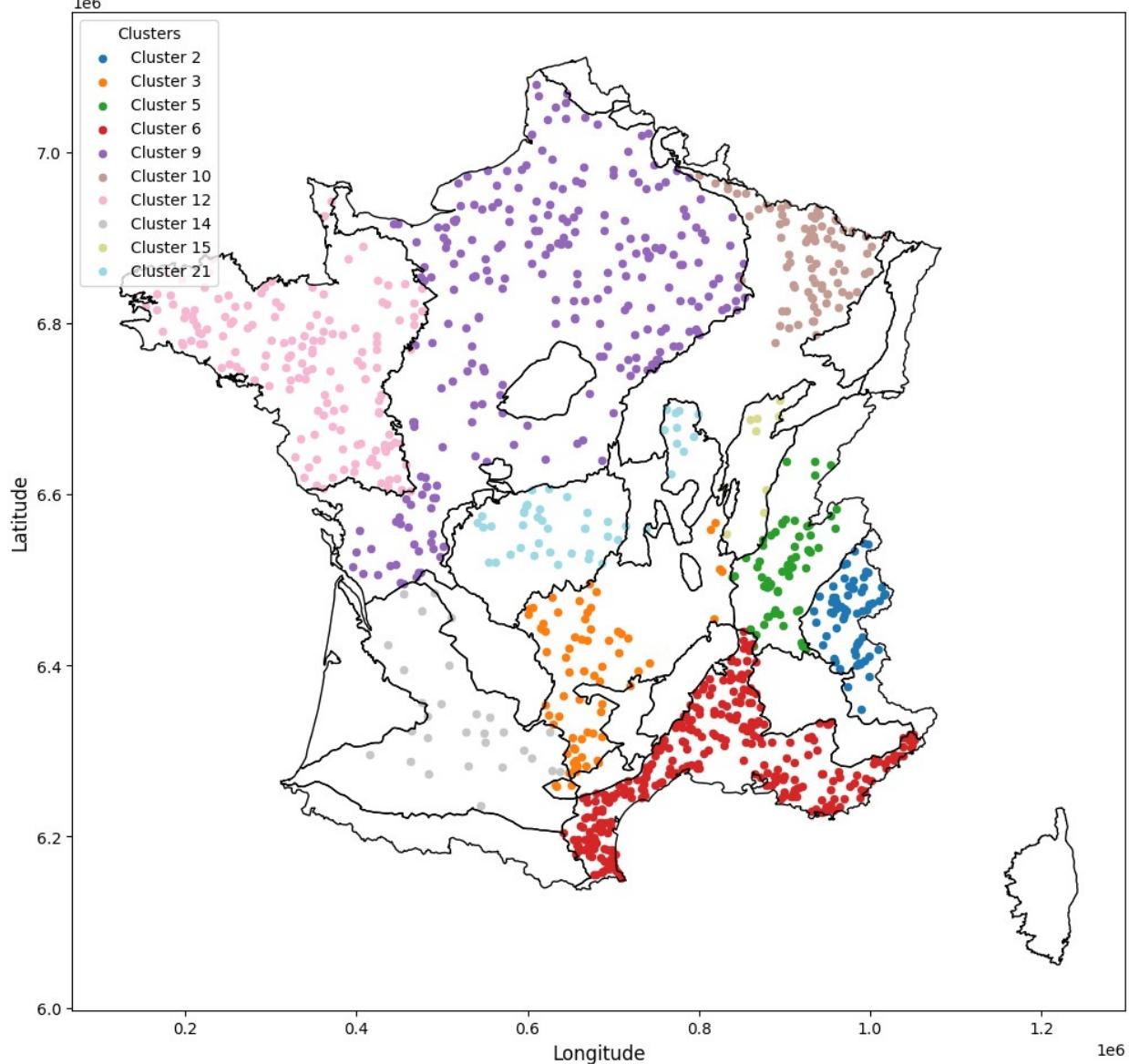
```
# Représentation des stations bien classées
correct_stations =
df_analyse_etude[df_analyse_etude['correct_classification']]
carto_correct_i2m2 = gpd.GeoDataFrame(correct_stations,
crs=crs_lambert,
geometry=gpd.GeoSeries(correct_stations.apply(lambda x:
geom.Point(x[x_col], x[y_col]), axis=1)))
HER_lambert = df_hydroregions.to_crs(crs_lambert)
unique_clusters_correct = carto_correct_i2m2[cluster_col].unique()
unique_clusters_correct = sorted(unique_clusters_correct)
cmap = cm.get_cmap('tab20', len(unique_clusters_correct))
cluster_colors_correct = {cluster: mcolors.to_hex(cmap(i)) for i,
```

```
cluster in enumerate(unique_clusters_correct)}  
fig, ax = plt.subplots(1, 1, figsize=(12, 10))  
HER_lambert.boundary.plot(ax=ax, color='black', linewidth=1)  
for cluster, color in cluster_colors_correct.items():  
    cluster_data = carto_correct_i2m2[carto_correct_i2m2[cluster_col]  
== cluster]  
    cluster_data.plot(ax=ax, color=color, markersize=20,  
label=f"Cluster {cluster}")  
ax.legend(loc='upper left', fontsize='medium', title='Clusters')  
plt.title('Stations correctement classées par cluster avec contours  
des Hydroécorégions', fontsize=16)  
plt.xlabel('Longitude', fontsize=12)  
plt.ylabel('Latitude', fontsize=12)  
plt.tight_layout()  
plt.show()
```

/var/folders/p1/4yqk4cyx3058m3j04rtkr56m0000gn/T/
ipykernel_53919/1846449924.py:8: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()`` or ``pyplot.get_cmap()`` instead.

Stations correctement classées par cluster avec contours des Hydroécorégions



Décalage temporel de 6 mois

Après avoir effectué le clustering avec un décalage temporel de 1 mois, les premiers résultats nous ont permis d'évaluer les regroupements d'hydroécostations en fonction des données physico-chimiques et biologiques. Pour approfondir cette analyse, nous allons explorer un décalage temporel plus important, de 6 mois, afin d'examiner si un lag différent peut améliorer la qualité des clusters, mieux refléter le délai entre les variations physico-chimiques et leurs répercussions biologiques, et enrichir notre compréhension des relations spatio-temporelles entre les hydroécostations.

```

df_pc_median_bio_median_6_month = pd.merge(df_pc_pivot_saison_median,
df_hydrobio_lag_6_month_median, on=['station', 'année', 'saison'],
how='inner')

df_pc_median_bio_median_6_month.shape

(35275, 20)

# Fusion avec le dataset des stations
df_pc_median_bio_median_6_month_with_coords = pd.merge(
    df_pc_median_bio_median_6_month,
    df_stations[['station', 'CoordXStationMesureEauxSurface',
    'CoordYStationMesureEauxSurface']],
    on='station',
    how='inner'
)

df_correl_6 = df_pc_median_bio_median_6_month_with_coords.copy()

# Ajout des hydrorégions pour voir si des caractéristiques y sont
corrélées
carto_i2m2_correl_6 = gpd.GeoDataFrame(df_correl_6, crs=crs_lambert,
geometry = gpd.GeoSeries(df_correl_6.agg(lambda
x:geom.Point(x.loc['CoordXStationMesureEauxSurface'],x.loc['CoordYStat
ionMesureEauxSurface']) ,axis=1)))
HER_stations_correl_6=carto_i2m2_correl_6.sjoin(df_hydroregions.to_crs
(crs_lambert),predicate='within').to_crs(crs_lambert)
df_correl_6 = HER_stations_correl_6.drop(columns=['geometry',
'index_right', 'NomHER1', 'gid'])
df_correl_6['saison'] = df_correl_6['saison'].factorize()[0]
# Mapping des colonnes pour pouvoir afficher la matrice de corrélation
correctement
column_mapping = {
    'Ammonium - Eau': 'NH4',
    'Azote Kjeldahl - Eau': 'NKjeldahl',
    'Carbone Organique - Eau': 'CO',
    'Conductivité à 25°C - Eau': 'Cond25',
    'Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau':
'DB05',
    'Matières en suspension - Eau': 'MES',
    'Nitrates - Eau': 'NO3',
    'Nitrites - Eau': 'NO2',
    'Orthophosphates (P04) - Eau': 'P04',
    'Oxygène dissous - Eau': 'O2',
    'Phosphore total - Eau': 'Ptot',
    'Potentiel en Hydrogène (pH) - Eau': 'pH',
    'Taux de saturation en oxygène - Eau': 'O2_sat',
    'Température de l\'Eau - Eau': 'TempEau',
    'Turbidité Formazine Néphéломétrique - Eau': 'Turbidité',
    'I2M2': 'I2M2',
}

```

```

    'CoordXStationMesureEauxSurface': 'CoordX',
    'CoordYStationMesureEauxSurface': 'CoordY'
}
df_correl_6 = df_correl_6.rename(columns=column_mapping,
index=column_mapping)

correlation_matrix_6 = df_correl_6.corr()
np.fill_diagonal(correlation_matrix_6.values, np.nan)
fig = px.imshow(correlation_matrix_6, text_auto=".1f",
color_continuous_scale='RdBu', title="Matrice de corrélation",
labels=dict(x="Caractéristiques", y="Caractéristiques",
color="Corrélation"), zmin=-1, zmax=1)
fig.update_layout(xaxis=dict(tickangle=45), autosize=True,
title_x=0.5, width=800, height=800)
fig.show()

{"config": {"plotlyServerURL": "https://plot.ly"}, "data":
[{"coloraxis": "coloraxis", "hovertemplate": "Caractéristiques: %
{x}<br>Caractéristiques: %{y}<br>Corrélation:
%{z}<br>", "name": "0", "texttemplate": "%
{z:.1f}", "type": "heatmap", "x": [
["station", "année", "saison", "NH4", "NKjeldahl", "C0", "Cond25", "DB05", "Di
uron -
Eau", "MES", "N03", "N02", "P04", "O2", "Ptot", "pH", "O2_sat", "TempEau", "Turb
idité", "I2M2", "CoordX", "CoordY", "CdHER1"], "xaxis": "x", "y": [
["station", "année", "saison", "NH4", "NKjeldahl", "C0", "Cond25", "DB05", "Di
uron -
Eau", "MES", "N03", "N02", "P04", "O2", "Ptot", "pH", "O2_sat", "TempEau", "Turb
idité", "I2M2", "CoordX", "CoordY", "CdHER1"], "yaxis": "y", "z": [[null, -0.17776238344480788, 0.12392926268274533, -1.406338471043048e-
2, 0.10556133304881198, -0.2761455057830036, 2.789576148834138e-2, -0.2427363493318209, 0.20784159225655088, -4.9374278619498134e-2, -0.36282048017563273, -0.103658569679514, -7.277983995390704e-
2, 3.077094649708927e-2, -0.11417390783462866, 0.23580923477694563, 0.15996618966183426, 8.20908062
1825272e-2, -2.683478468291022e-2, -1.7760013102271776e-
2, 0.279865481413387, -0.6532872221416204, -0.28535326481453516], [-0.17776238344480788, null, -2.29791341475085e-2, -9.135375964830843e-2, -0.4151646667652803, 0.1077852901858123, -2.3910674409038615e-2, -4.9611640954287085e-2, -0.11752248258943816, -2.1212349043993307e-
2, 0.15876465566170794, -3.397013604854253e-4, 2.444132208164581e-2, -2.578180696344993e-2, -1.7425596218461775e-2, -8.36352148833461e-2, -3.42668822311679e-2, 4.408139977523851e-2, 4.575224049092308e-
3, 1.91396645736207e-2, -0.16401173425341578, 0.1850869862910558, 0.10119781116288366], [0.12392926268274533, -2.29791341475085e-2, null, -5.365315581983148e-
2, 6.640755992729678e-4, -0.13521917612321485, -2.6076412521323294e-2, -0.1061890568614298, 4.841725319280723e-2, 7.022628901713165e-2, -0.1897009465233502, -3.757460993916908e-2, -4.185436203847084e-2, -0.4093259062419663, -3.0663848240129583e-2, 8.47770853359883e-
2]
}

```

2, 1.3545548804826339e-3, 0.5523579921051074, 3.073902666021203e-
 2, 6.562316069546253e-2, 8.944340113454298e-2, -2.6351724974401386e-2, -
 0.1710413520105562], [-1.406338471043048e-2, -9.135375964830843e-2, -
 5.365315581983148e-
 2, null, 0.44912240559785577, 0.14302761814796633, 0.1786434916077149, 0.29
 59808872815493, 6.918172763297523e-2, 5.326582991829478e-
 2, 8.314306791482e-2, 0.49842636947456687, 0.39360410325193973, -
 0.13474782716537165, 0.41261167633544993, 6.46110117701733e-3, -
 0.17544003311927128, -2.339735931369468e-2, 6.581773676527619e-2, -
 0.2536301508052299, 1.7524169200149306e-2, 3.722946419879074e-
 2, 3.110712409400671e-2], [0.10556133304881198, -
 0.4151646667652803, 6.640755992729678e-
 4, 0.44912240559785577, null, 0.311198259725792, 9.196158269852024e-
 2, 0.3147128863748662, 4.9120593204007246e-
 2, 0.2759277551172183, 7.288303678927113e-
 3, 0.3184474295472012, 0.2694066758580303, -
 0.10947763537958961, 0.41488422039355105, 2.9594741888167844e-3, -
 0.14529280172214953, 3.567431282176698e-3, 0.3489141778147033, -
 0.2291503584308481, 4.743631911310915e-3, -3.65814820009697e-
 2, 2.8391417291562415e-2], [-0.2761455057830036, 0.1077852901858123, -
 0.13521917612321485, 0.14302761814796633, 0.311198259725792, null, -
 4.4423208336714856e-2, 0.4003667922001568, -
 0.11123112854851405, 0.19268701960813614, 0.25683832820615476, 0.28639031
 333504295, 0.26525272115155885, -
 0.13358233952596382, 0.3788534139612309, -0.3269090782759772, -
 0.2671935319971638, -1.3725187034714837e-2, 0.22051511527870338, -
 0.2281497140187007, -
 0.4045146040298428, 0.25658472342582545, 0.3709877194302665],
 [2.789576148834138e-2, -2.3910674409038615e-2, -2.6076412521323294e-
 2, 0.1786434916077149, 9.196158269852024e-2, -4.4423208336714856e-
 2, null, 2.6809236621544334e-2, 0.1228952368880075, 4.620078754090977e-
 2, 0.3915744660015941, 0.3580405689826878, 0.3276308008124963, -
 0.19985347652651198, 0.2669195231612198, 0.5062387679654877, -
 0.17631658800855216, 0.1981279689971131, 4.4246805499070056e-2, -
 0.49172746176882515, 0.11074116471593176, 8.064892728369863e-2, -
 6.022961236907024e-2], [-0.2427363493318209, -4.9611640954287085e-2, -
 0.1061890568614298, 0.2959808872815493, 0.3147128863748662, 0.40036679220
 01568, 2.6809236621544334e-2, null, 1.2833729943724712e-
 2, 0.21352336713056358, 0.1546600839072463, 0.36110828531355366, 0.2850308
 02161658, -9.25528273963027e-2, 0.4307540621469691, -
 0.10821095651863788, -0.17369806042706618, -4.4307092026515504e-
 2, 0.20712815018033834, -0.22093019208013667, -
 0.13914359086235364, 0.21242845260177437, 0.20285713670905448],
 [0.20784159225655088, -0.11752248258943816, 4.841725319280723e-
 2, 6.918172763297523e-2, 4.9120593204007246e-2, -
 0.11123112854851405, 0.1228952368880075, 1.2833729943724712e-
 2, null, 3.0403776075986814e-2, -7.904434186356064e-
 2, 0.1001168102407422, 0.12783345247112826, -1.3906886414004165e-
 2, 0.13798550358316802, 0.13357980268624806, 3.6426152578112624e-

2, 6.0565235794372015e-2, 4.946129126002056e-2, -
 0.11848158413023453, 0.18628421155839062, -0.14255062400199656, -
 6.914554163283189e-2], [-4.9374278619498134e-2, -2.1212349043993307e-
 2, 7.022628901713165e-2, 5.326582991829478e-
 2, 0.2759277551172183, 0.19268701960813614, 4.620078754090977e-
 2, 0.21352336713056358, 3.0403776075986814e-
 2, null, 0.10239806784415402, 9.407532921857453e-2, 0.11291581655203094, -
 7.494760965534553e-2, 0.2979454629245437, 2.366803526015744e-2, -
 6.032478044759189e-2, 4.695315274411401e-2, 0.7358989962218027, -
 0.11867748686652156, -3.3148055213339314e-2, 8.990817946679738e-
 2, 6.332805444394012e-2], [-0.36282048017563273, 0.15876465566170794, -
 0.1897009465233502, 8.314306791482e-2, 7.288303678927113e-
 3, 0.25683832820615476, 0.3915744660015941, 0.1546600839072463, -
 7.904434186356064e-
 2, 0.10239806784415402, null, 0.3492738450804181, 0.2257381511567851, -
 0.10940011873792883, 0.22694214834759704, 3.0704996900828552e-2, -
 0.21495878132676924, 8.273151960058525e-2, 9.748054635858229e-2, -
 0.2699832011697142, -
 0.36715330768783927, 0.3976544070861012, 0.24158368422712617], [-
 0.103658569679514, -3.397013604854253e-4, -3.757460993916908e-
 2, 0.49842636947456687, 0.3184474295472012, 0.28639031333504295, 0.3580405
 689826878, 0.36110828531355366, 0.1001168102407422, 9.407532921857453e-
 2, 0.3492738450804181, null, 0.5526306498873845, -
 0.2258322113449765, 0.5572446306022584, 5.911149359240028e-2, -
 0.26853157609252304, 0.10064897754045718, 0.11871841031533555, -
 0.42662275432137, -5.2536230555160233e-
 2, 0.1484752489814761, 0.14482054379201148], [-7.277983995390704e-
 2, 2.444132208164581e-2, -4.185436203847084e-
 2, 0.39360410325193973, 0.2694066758580303, 0.26525272115155885, 0.3276308
 008124963, 0.285030802161658, 0.12783345247112826, 0.11291581655203094, 0.
 2257381511567851, 0.5526306498873845, null, -
 0.170251018324847, 0.7868817107946651, 6.130748531427216e-2, -
 0.212900333424042, 6.34603661728118e-2, 0.15188974329099658, -
 0.3738194132511054, 4.621738247665473e-
 3, 0.11310834474645533, 8.27104216553219e-2], [3.077094649708927e-2, -
 2.578180696344993e-2, -0.4093259062419663, -0.13474782716537165, -
 0.10947763537958961, -0.13358233952596382, -0.19985347652651198, -
 9.25528273963027e-2, -1.3906886414004165e-2, -7.494760965534553e-2, -
 0.10940011873792883, -0.2258322113449765, -0.170251018324847, null, -
 0.2252784890714889, 0.11786505156169219, 0.6803546790753503, -
 0.6047614368089607, -8.937743220972229e-
 2, 0.19157790130404925, 0.1619286751370773, -1.1208676257726962e-2, -
 4.877021289453231e-2], [-0.11417390783462866, -1.7425596218461775e-2, -
 3.0663848240129583e-
 2, 0.41261167633544993, 0.41488422039355105, 0.3788534139612309, 0.2669195
 231612198, 0.4307540621469691, 0.13798550358316802, 0.2979454629245437, 0.
 22694214834759704, 0.5572446306022584, 0.7868817107946651, -
 0.2252784890714889, null, 5.742911942927795e-3, -
 0.25660209873740547, 6.728471540542669e-2, 0.3355685490071149, -

```

0.3771186226699921,-4.319862588997806e-
2,0.14451342003087367,0.13869690291304604],[0.23580923477694563,-
8.36352148833461e-2,8.47770853359883e-2,6.46110117701733e-
3,2.9594741888167844e-3,-0.3269090782759772,0.5062387679654877,-
0.10821095651863788,0.13357980268624806,2.366803526015744e-
2,3.0704996900828552e-2,5.911149359240028e-2,6.130748531427216e-
2,0.11786505156169219,5.742911942927795e-
3,null,0.21290945417587911,7.043581362501415e-2,2.834586801370292e-2,-
0.15335797703811832,0.2729065820420163,-9.652917358294542e-2,-
0.242900507073053],[0.15996618966183426,-3.42668822311679e-
2,1.3545548804826339e-3,-0.17544003311927128,-0.14529280172214953,-
0.2671935319971638,-0.17631658800855216,-
0.17369806042706618,3.6426152578112624e-2,-6.032478044759189e-2,-
0.21495878132676924,-0.26853157609252304,-
0.212900333424042,0.6803546790753503,-
0.25660209873740547,0.21290945417587911,null,-8.894263471047303e-2,-
8.643770046020273e-2,0.2520598775878592,0.1747814072844321,-
0.13199445168480808,-0.16284677391977256],[8.209080621825272e-
2,4.408139977523851e-2,0.5523579921051074,-2.339735931369468e-
2,3.567431282176698e-3,-1.3725187034714837e-2,0.1981279689971131,-
4.4307092026515504e-2,6.0565235794372015e-2,4.695315274411401e-
2,8.273151960058525e-2,0.10064897754045718,6.34603661728118e-2,-
0.6047614368089607,6.728471540542669e-2,7.043581362501415e-2,-
8.894263471047303e-2,null,4.836558115074987e-2,-8.486157603752642e-2,-
0.17791275859004896,-8.183523703896058e-2,2.498858497232908e-2],[-
2.683478468291022e-2,4.575224049092308e-3,3.073902666021203e-
2,6.581773676527619e-
2,0.3489141778147033,0.22051511527870338,4.4246805499070056e-
2,0.20712815018033834,4.946129126002056e-
2,0.7358989962218027,9.748054635858229e-
2,0.11871841031533555,0.15188974329099658,-8.937743220972229e-
2,0.3355685490071149,2.834586801370292e-2,-8.643770046020273e-
2,4.836558115074987e-2,null,-0.13130422025851238,-2.7614683451160003e-
2,7.38725872907628e-2,8.876041837601371e-2],[-1.7760013102271776e-
2,1.91396645736207e-2,6.562316069546253e-2,-0.2536301508052299,-
0.2291503584308481,-0.2281497140187007,-0.49172746176882515,-
0.22093019208013667,-0.11848158413023453,-0.11867748686652156,-
0.2699832011697142,-0.42662275432137,-
0.3738194132511054,0.19157790130404925,-0.3771186226699921,-
0.15335797703811832,0.2520598775878592,-8.486157603752642e-2,-
0.13130422025851238,null,-6.641415014902961e-2,-0.1380917101226196,-
7.329216784066604e-2],[0.279865481413387,-
0.16401173425341578,8.944340113454298e-2,1.7524169200149306e-
2,4.743631911310915e-3,-0.4045146040298428,0.11074116471593176,-
0.13914359086235364,0.18628421155839062,-3.3148055213339314e-2,-
0.36715330768783927,-5.2536230555160233e-2,4.621738247665473e-
3,0.1619286751370773,-4.319862588997806e-
2,0.2729065820420163,0.1747814072844321,-0.17791275859004896,-
2.7614683451160003e-2,-6.641415014902961e-2,null,-5.245417768462823e-

```

```

2,-0.17279769247411458],[-0.6532872221416204,0.1850869862910558,-
2.6351724974401386e-2,3.722946419879074e-2,-3.65814820009697e-
2,0.25658472342582545,8.064892728369863e-2,0.21242845260177437,-
0.14255062400199656,8.990817946679738e-
2,0.3976544070861012,0.1484752489814761,0.11310834474645533,-
1.1208676257726962e-2,0.14451342003087367,-9.652917358294542e-2,-
0.13199445168480808,-8.183523703896058e-2,7.38725872907628e-2,-
0.1380917101226196,-5.245417768462823e-2,null,0.20095186224474407],[-
0.28535326481453516,0.10119781116288366,-
0.1710413520105562,3.110712409400671e-2,2.8391417291562415e-
2,0.3709877194302665,-6.022961236907024e-2,0.20285713670905448,-
6.914554163283189e-2,6.332805444394012e-
2,0.24158368422712617,0.14482054379201148,8.27104216553219e-2,-
4.877021289453231e-2,0.13869690291304604,-0.242900507073053,-
0.16284677391977256,2.498858497232908e-2,8.876041837601371e-2,-
7.329216784066604e-2,-
0.17279769247411458,0.20095186224474407,null]]}],"layout": {
    "autosize": true,
    "coloraxis": {"cmax": 1, "cmin": -1, "colorbar": {"title": {"text": "Corrélation"}}, "colorscale": [[0, "rgb(103,0,31)", [0.1, "rgb(178,24,43)"], [0.2, "rgb(214,96,77)"], [0.3, "rgb(244,165,130)"], [0.4, "rgb(253,219,199)"], [0.5, "rgb(247,247,247)"], [0.6, "rgb(209,229,240)"], [0.7, "rgb(146,197,222)"], [0.8, "rgb(67,147,195)"], [0.9, "rgb(33,102,172)"]], [1, "rgb(5,48,97)"]]}, "height": 800, "template": {"data": {"bar": [{"error_x": {"color": "#2a3f5f"}, "error_y": {"color": "#2a3f5f"}, "marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}, "type": "bar"}], "barpolar": [{"marker": {"line": {"color": "#E5ECF6", "width": 0.5}, "pattern": {"fillmode": "overlay", "size": 10, "solidity": 0.2}}, "type": "barpolar"}], "carpet": [{"aaxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "baxis": {"endlinecolor": "#2a3f5f", "gridcolor": "white", "linecolor": "white", "minorgridcolor": "white", "startlinecolor": "#2a3f5f"}, "type": "carpet"}], "choropleth": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "choropleth"}], "contour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}, {"type": "contour"}], "contourcarpet": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "contourcarpet"}], "heatmap": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"]]}]}]}]
```

```

[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],
[1, "#f0f921]], "type": "heatmap"}], "heatmapgl": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], "histogram": [{"marker": {"pattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}}, "type": "histogram"}], "histogram2d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], "histogram2dcontour": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]], "mesh3d": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "type": "mesh3d"}, {"parcoords": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}, "type": "parcoords"}], "pie": [{"autoMargin": true, "type": "pie"}], "scatter": [{"fillPattern": {"fillMode": "overlay", "size": 10, "solidity": 0.2}, "type": "scatter"}], "scatter3d": [{"line": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatter3d"}], "scattercarpet": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattercarpet"}], "scattergeo": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergeo"}], "scattergl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattergl"}], "scattermapbox": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scattermapbox"}], "scatterpolar": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolar"}], "scatterpolargl": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterpolargl"}], "scatterternary": [{"marker": {"colorbar": {"outlineWidth": 0, "ticks": ""}}}, {"type": "scatterternary"}], "surface": [{"colorbar": {"outlineWidth": 0, "ticks": ""}, "colorscale": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"]]}]]}

```

```

[0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"],
[0.8888888888888888, "#fdca26"],
[1, "#f0f921"], {"type": "surface"}], "table": [{"cells": {"fill": {"color": "#EBF0F8"}, "line": {"color": "white"}}, "header": {"fill": {"color": "#C8D4E3"}, "line": {"color": "white"}}, {"type": "table"}}], "layout": {"annotationdefaults": {"arrowcolor": "#2a3f5f", "arrowhead": 0, "arrowwidth": 1}, "autotypenumbers": "strict", "coloraxis": {"colorbar": {"outlinewidth": 0, "ticks": ""}}, "colorscale": {"diverging": [[0, "#8e0152"], [0.1, "#c51b7d"], [0.2, "#de77ae"], [0.3, "#f1b6da"], [0.4, "#fde0ef"], [0.5, "#f7f7f7"], [0.6, "#e6f5d0"], [0.7, "#b8e186"], [0.8, "#7fbc41"], [0.9, "#4d9221"], [1, "#276419"]], "sequential": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]], "sequentialminus": [[0, "#0d0887"], [0.1111111111111111, "#46039f"], [0.2222222222222222, "#7201a8"], [0.3333333333333333, "#9c179e"], [0.4444444444444444, "#bd3786"], [0.5555555555555556, "#d8576b"], [0.6666666666666666, "#ed7953"], [0.7777777777777778, "#fb9f3a"], [0.8888888888888888, "#fdca26"], [1, "#f0f921"]]}, "colorway": [{"#636efa", "#EF553B", "#00cc96", "#ab63fa", "#FFA15A", "#19d3f3", "#FF6692", "#B6E880", "#FF97FF", "#FECB52"}, {"font": {"color": "#2a3f5f"}, "geo": {"bgcolor": "white", "lakecolor": "white", "landcolor": "#E5ECF6", "showlake": true, "showland": true, "subunitcolor": "white"}, "hoverlabel": {"align": "left"}, "hovermode": "closest", "mapbox": {"style": "light"}, "paper_bgcolor": "white", "plot_bgcolor": "#E5ECF6", "polar": {"angularaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6", "radialaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, "scene": {"xaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "yaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}, "zaxis": {"backgroundcolor": "#E5ECF6", "gridcolor": "white", "gridwidth": 2, "linecolor": "white", "showbackground": true, "ticks": "", "zerolinecolor": "white"}}, {"shapedefaults": {"line": {"color": "#2a3f5f"}}, "ternary": {"aaxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "baxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}, "bgcolor": "#E5ECF6"}, {"caxis": {"gridcolor": "white", "linecolor": "white", "ticks": ""}}, {"title": {"x": 5.0e-2}, "xaxis": {"automargin": true, "gridcolor": "white", "linecolor": "white", "ticks": ""}, "title": ""]}], "type": "table"}]

```

```

{"standoff":15}, "zerolinecolor": "white", "zerolinewidth":2}, "yaxis": {"automargin":true, "gridcolor": "white", "linecolor": "white", "ticks": "", "title": {"standoff":15}, "zerolinecolor": "white", "zerolinewidth":2}}}, "title": {"text": "Matrice de corrélation", "x":0.5}, "width":800, "xaxis": {"anchor": "y", "constrain": "domain", "domain": [0,1], "scaleanchor": "y", "tickangle":45, "title": {"text": "Caractéristiques"}}, "yaxis": {"anchor": "x", "autorange": "reversed", "constrain": "domain", "domain": [0,1], "title": {"text": "Caractéristiques"}}}}

```

Les mêmes observations sont présentes qu'auparavant : **I2M2** reste négativement corrélé avec Cond25, NO2, PO et Ptot, et ce, malgré le décalage de 6 mois. Cela suggère que ces paramètres sont probablement les principaux facteurs influençant l'état hydrobiologique de l'eau, indépendamment de l'impact des saisons.

Clustering pour 6 mois de lag

```

df_pc_median_bio_median_6_month_with_coords.isnull().sum()

station 0
année 0
saison 0
Ammonium - Eau 1942
Azote Kjeldahl - Eau 2117
Carbone Organique - Eau 1602
Conductivité à 25°C - Eau 195
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 1843
Diuron - Eau 7569
Matières en suspension - Eau 382
Nitrates - Eau 1791
Nitrites - Eau 1803
Orthophosphates (P04) - Eau 1784
Oxygène dissous - Eau 199
Phosphore total - Eau 1491
Potentiel en Hydrogène (pH) - Eau 190
Taux de saturation en oxygène - Eau 1126
Température de l'Eau - Eau 139
Turbidité Formazine Néphéломétrique - Eau 6503
I2M2 0
CoordXStationMesureEauxSurface 0
CoordYStationMesureEauxSurface 0
dtype: int64

missing_percentage_6 =
df_pc_median_bio_median_6_month_with_coords.isnull().mean() * 100
print(missing_percentage_6)

station 0.000000
année 0.000000

```

```

saison                               0.000000
Ammonium - Eau                      9.141835
Azote Kjeldahl - Eau                 9.965636
Carbone Organique - Eau              7.541308
Conductivité à 25°C - Eau           0.917949
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 8.675799
Diuron - Eau                         35.630561
Matières en suspension - Eau        1.798239
Nitrates - Eau                       8.431013
Nitrites - Eau                       8.487502
Orthophosphates (P04) - Eau         8.398061
Oxygène dissous - Eau                0.936779
Phosphore total - Eau               7.018783
Potentiel en Hydrogène (pH) - Eau   0.894412
Taux de saturation en oxygène - Eau 5.300570
Température de l'Eau - Eau          0.654333
Turbidité Formazine Néphéломétrique - Eau 30.612437
I2M2                                 0.000000
CoordXStationMesureEauxSurface     0.000000
CoordYStationMesureEauxSurface      0.000000
dtype: float64

```

```

# on supprime aussi Diuron - Eau
df_pc_median_bio_median_6_month_with_coords.drop(columns=['Diuron - Eau'], inplace=True)

# Calculer le nombre de valeurs manquantes par ligne
missing_values_per_row_6 =
df_pc_median_bio_median_6_month_with_coords.isnull().sum(axis=1)
missing_summary_6 =
missing_values_per_row_6.value_counts().reset_index()
missing_summary_6.columns = ['Nombre de valeurs manquantes', 'Nombre de lignes']
print(missing_summary_6.sort_values('Nombre de valeurs manquantes'))

```

	Nombre de valeurs manquantes	Nombre de lignes
0	0	11844
1	1	6833
3	2	552
7	3	137
9	4	64
5	5	204
6	6	157
8	7	133
2	8	903
10	9	47
4	10	277
12	11	34
14	12	1
15	13	1

11	14	46
13	15	10

```

# Imputation des valeurs manquantes par la médiane
imputer = SimpleImputer(strategy='median') # ici on pourrait aussi mettre mean
colonnes = df_pc_median_bio_median_6_month_with_coords.columns
colonnes = colonnes.drop('station')
colonnes = colonnes.drop('année')
colonnes = colonnes.drop('saison')
colonnes = colonnes.drop('I2M2')
df_pc_median_bio_median_6_month_with_coords[colonnes] =
imputer.fit_transform(df_pc_median_bio_median_6_month_with_coords[colonnes])
df_pc_median_bio_median_6_month_with_coords.isnull().sum()

station 0
année 0
saison 0
Ammonium - Eau 0
Azote Kjeldahl - Eau 0
Carbone Organique - Eau 0
Conductivité à 25°C - Eau 0
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau 0
Matières en suspension - Eau 0
Nitrates - Eau 0
Nitrites - Eau 0
Orthophosphates (P04) - Eau 0
Oxygène dissous - Eau 0
Phosphore total - Eau 0
Potentiel en Hydrogène (pH) - Eau 0
Taux de saturation en oxygène - Eau 0
Température de l'Eau - Eau 0
Turbidité Formazine Néphéломétrique - Eau 0
I2M2 0
CoordXStationMesureEauxSurface 0
CoordYStationMesureEauxSurface 0
dtype: int64

df_pc_median_bio_median_6_month_with_coords_for_regression =
df_pc_median_bio_median_6_month_with_coords.copy()

# Mapper chaque saison à un trimestre
saison_to_quarter = {
    "Hiver": 1,
    "Printemps": 2,
    "Été": 3,
    "Automne": 4
}
df_pc_median_bio_median_6_month_with_coords['trimestre'] =

```

```

df_pc_median_bio_median_6_month_with_coords['saison'].map(saison_to_quarter)
# drop saison
df_pc_median_bio_median_6_month_with_coords.drop(columns=['saison'],
inplace=True)
df_pc_median_bio_median_6_month_with_coords.head(5)

    station   année   Ammonium - Eau   Azote Kjeldahl - Eau \
0  5001800  2007        0.04                  1.0
1  5001800  2008        0.04                  1.0
2  5001800  2009        0.06                  1.0
3  5005350  2007        0.12                  1.0
4  5005350  2008        0.04                  1.0

    Carbone Organique - Eau   Conductivité à 25°C - Eau \
0                      6.60            895.0
1                      5.50            803.5
2                      4.60            833.5
3                      1.65            593.0
4                      1.50            632.5

    Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \
0                      0.50
1                      0.65
2                      0.50
3                      0.90
4                      1.60

    Matières en suspension - Eau   Nitrates - Eau   Nitrites -
Eau ... \
0                   11.0          45.60         0.070 ...
1                   7.5           39.25         0.070 ...
2                  10.0          41.20         0.060 ...
3                  15.0          45.40         0.135 ...
4                  21.0          44.40         0.050 ...

    Oxygène dissous - Eau   Phosphore total - Eau \
0                   11.05          0.05
1                   9.15          0.05
2                  10.70          0.06
3                   8.30          0.16
4                  10.35          0.13

    Potentiel en Hydrogène (pH) - Eau   Taux de saturation en oxygène -
Eau \

```

```

0                               7.75
100.5
1                               7.85
90.0
2                               7.95
91.5
3                               7.90
80.0
4                               8.00
95.0

    Température de l'Eau - Eau   Turbidité Formazine Néphéломétrique -
Eau \
0                               11.35
5.0
1                               12.85
5.0
2                               8.25
5.0
3                               11.80
5.0
4                               10.70
5.0

    I2M2  CoordXStationMesureEauxSurface
CoordYStationMesureEauxSurface \
0  0.3004                      399856.0
6531980.0
1  0.3848                      399856.0
6531980.0
2  0.5756                      399856.0
6531980.0
3  0.4851                      450151.0
6572920.0
4  0.3488                      450151.0
6572920.0

    trimestre
0      1
1      2
2      1
3      2
4      1

[5 rows x 21 columns]

# Convertir l'identifiant de station en variable catégorielle (cela
crée un encodage numérique)
df_pc_median_bio_median_6_month_with_coords['station'] =
df_pc_median_bio_median_6_month_with_coords['station'].astype(str)

```

```

df_pc_median_bio_median_6_month_with_coords['trimestre'] =
df_pc_median_bio_median_6_month_with_coords['trimestre'].astype(int)
df_pc_median_bio_median_6_month_with_coords['année'] =
df_pc_median_bio_median_6_month_with_coords['année'].astype(int)
df_pc_median_bio_median_6_month_with_coords.dtypes

station                                         object
année                                         int64
Ammonium - Eau                                float64
Azote Kjeldahl - Eau                            float64
Carbone Organique - Eau                          float64
Conductivité à 25°C - Eau                      float64
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau float64
Matières en suspension - Eau                   float64
Nitrates - Eau                                  float64
Nitrites - Eau                                  float64
Orthophosphates (P04) - Eau                   float64
Oxygène dissous - Eau                           float64
Phosphore total - Eau                          float64
Potentiel en Hydrogène (pH) - Eau              float64
Taux de saturation en oxygène - Eau            float64
Température de l'Eau - Eau                     float64
Turbidité Formazine Néphéломétrique - Eau    float64
I2M2                                           float64
CoordXStationMesureEauxSurface                float64
CoordYStationMesureEauxSurface                float64
trimestre                                      int64
dtype: object

# drop les coordonnées
df_pc_median_bio_median_6_month_c =
df_pc_median_bio_median_6_month_with_coords.copy()
df_pc_median_bio_median_6_month_c =
df_pc_median_bio_median_6_month_with_coords.drop(columns=['CoordXStationMesureEauxSurface', 'CoordYStationMesureEauxSurface'])
df_clustering_6 = df_pc_median_bio_median_6_month_c.copy()

# Encodage des stations
le = LabelEncoder()
df_clustering_6['station_encoded'] =
le.fit_transform(df_clustering_6['station'])
df_clustering_6.drop(columns=['station'], inplace=True)
# Normalisation des données pour qu'elles aient toutes la même
importance
scaler = MinMaxScaler()
normalized_data_6 = scaler.fit_transform(df_clustering_6)
normalized_data_6 = pd.DataFrame(normalized_data_6,
columns=df_clustering_6.columns)

```

On teste ici directement avec 22 clusters, qui est le nombre d'hydroécroégions.

```

kmeans = KMeans(n_clusters=22, random_state=42)
kmeans.fit(normalized_data_6)

KMeans(n_clusters=22, random_state=42)

# Analyse des centres des clusters
cluster_centers_6 = pd.DataFrame(kmeans.cluster_centers_,
columns=df_clustering_6.columns)
print("Centres des clusters :")
print(cluster_centers_6)
# Variabilité des caractéristiques par cluster
influence_6 = cluster_centers_6.std(axis=0)
print("\nInfluence des caractéristiques :")
print(influence_6.sort_values(ascending=False))

Centres des clusters :
    année Ammonium - Eau Azote Kjeldahl - Eau Carbone Organique
- Eau \
0  0.698467          0.003703          0.103479
0.090523
1  0.328105          0.015226          0.141999
0.165752
2  0.485039          0.153096          0.295034
0.259995
3  0.726435          0.006011          0.106410
0.155199
4  0.295109          0.009563          0.167491
0.111694
5  0.728114          0.003733          0.102967
0.053176
6  0.765597          0.007173          0.108304
0.156238
7  0.565142          0.007076          0.108252
0.128290
8  0.622577          0.016792          0.143201
0.164365
9  0.225716          0.010606          0.196300
0.060238
10 0.718125          0.026329          0.130423
0.197367
11 0.797656          0.008145          0.120576
0.217816
12 0.290601          0.014211          0.171691
0.186506
13 0.657866          0.019224          0.209163
0.484424
14 0.886146          0.011175          0.127999
0.216051
15 0.193605          0.009852          0.177127
0.134025

```

16	0.809513 0.179006	0.005938	0.107138
17	0.791328 0.151644	0.014402	0.118397
18	0.730698 0.168549	0.012129	0.135354
19	0.833558 0.079183	0.005049	0.101595
20	0.301445 0.084360	0.009649	0.185023
21	0.281307 0.165406	0.024189	0.198796

Conductivité à 25°C - Eau \

0	0.088203
1	0.235228
2	0.464355
3	0.125860
4	0.169859
5	0.188359
6	0.291556
7	0.187297
8	0.277654
9	0.238012
10	0.398769
11	0.249009
12	0.352056
13	0.159425
14	0.221966
15	0.124263
16	0.073163
17	0.368012
18	0.305563
19	0.268921
20	0.234057
21	0.306888

Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \

0	0.065298
1	0.130716
2	0.193447
3	0.079933
4	0.071575
5	0.045762
6	0.077588
7	0.074638
8	0.104098
9	0.047029
10	0.115470

11		0.078397
12		0.086406
13		0.133124
14		0.088208
15		0.088291
16		0.072097
17		0.082495
18		0.071578
19		0.061643
20		0.050383
21		0.096025

	Matières en suspension - Eau	Nitrates - Eau	Nitrites - Eau \
0	0.008385	0.044672	0.009643
1	0.026817	0.177814	0.045239
2	0.044412	0.283239	0.302273
3	0.020160	0.141737	0.023936
4	0.023763	0.084879	0.029409
5	0.043159	0.042185	0.012995
6	0.031723	0.531703	0.050400
7	0.019968	0.163281	0.023793
8	0.027181	0.183589	0.087617
9	0.051094	0.048593	0.026538
10	0.039794	0.437752	0.123712
11	0.016723	0.233073	0.047867
12	0.039792	0.558818	0.074136
13	0.052333	0.330349	0.079309
14	0.042025	0.183130	0.042439
15	0.017284	0.127408	0.028544
16	0.020638	0.131848	0.017043
17	0.024271	0.312801	0.082772
18	0.024801	0.160138	0.065307
19	0.012963	0.090258	0.017181
20	0.020750	0.073336	0.023061
21	0.038468	0.199149	0.067141

	Orthophosphates (P04) - Eau	Oxygène dissous - Eau	Phosphore total - Eau \
0	0.017070	0.678282	0.010015
1	0.056833	0.646732	0.063032
2	0.385383	0.526487	0.291235
3	0.027735	0.611171	0.025080
4	0.022256	0.609590	0.021344
5	0.016963	0.608848	

0.018037		
6	0.047730	0.630103
0.037141		
7	0.029377	0.655852
0.025800		
8	0.073168	0.593819
0.060626		
9	0.021058	0.576355
0.022356		
10	0.109079	0.622717
0.080897		
11	0.063266	0.486264
0.047855		
12	0.052693	0.618875
0.049519		
13	0.070075	0.614420
0.076135		
14	0.046884	0.614443
0.044560		
15	0.030029	0.665710
0.025451		
16	0.027044	0.674340
0.023953		
17	0.085515	0.633999
0.056421		
18	0.103373	0.532261
0.075265		
19	0.026563	0.667578
0.015353		
20	0.026028	0.668148
0.023482		
21	0.074703	0.652616
0.058557		

Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène - Eau \		
0	0.534866	
0.577948		
1	0.582482	
0.556129		
2	0.614813	
0.417885		
3	0.550950	
0.576189		
4	0.614104	
0.575194		
5	0.660344	
0.610470		
6	0.605197	

0.535294	
7	0.603768
0.557894	
8	0.621437
0.558796	
9	0.638696
0.552772	
10	0.620540
0.504520	
11	0.526159
0.414533	
12	0.634470
0.517792	
13	0.474777
0.496726	
14	0.556240
0.503130	
15	0.543336
0.555357	
16	0.461223
0.565405	
17	0.641237
0.534562	
18	0.614878
0.492453	
19	0.671149
0.584120	
20	0.638021
0.567288	
21	0.625705
0.543336	

Température de l'Eau - Eau	Turbidité Formazine Néphélométrique -
Eau \	
0	0.320364
0.009095	
1	0.326368
0.023243	
2	0.433543
0.026460	
3	0.494306
0.017761	
4	0.485099
0.013905	
5	0.478646
0.039370	
6	0.419217
0.027180	
7	0.355174

0.016537	
8	0.531466
0.026199	
9	0.509798
0.015191	
10	0.368822
0.034634	
11	0.548253
0.016272	
12	0.414184
0.018444	
13	0.399058
0.054600	
14	0.405212
0.041963	
15	0.324421
0.012678	
16	0.322893
0.018696	
17	0.395457
0.026785	
18	0.562344
0.028084	
19	0.368494
0.015001	
20	0.357372
0.016713	
21	0.362369
0.018072	

	I2M2	trimestre	station_encoded
0	0.786771	2.145594e-02	0.828852
1	0.552620	4.575163e-02	0.065148
2	0.225474	4.724409e-02	0.744078
3	0.757723	3.368580e-01	0.379359
4	0.693742	3.333333e-01	0.644466
5	0.685469	4.958791e-01	0.778428
6	0.653733	6.377176e-02	0.249745
7	0.670405	-3.330669e-16	0.519138
8	0.356842	3.327423e-01	0.680581
9	0.590291	8.312005e-01	0.742782
10	0.256164	4.097453e-02	0.115745
11	0.494730	9.978022e-01	0.300571
12	0.474140	4.962055e-02	0.513581
13	0.495120	2.764798e-02	0.358383
14	0.376724	2.340506e-02	0.337113
15	0.698648	6.033183e-04	0.527296
16	0.740751	1.510004e-03	0.318225
17	0.267692	3.523035e-03	0.765193

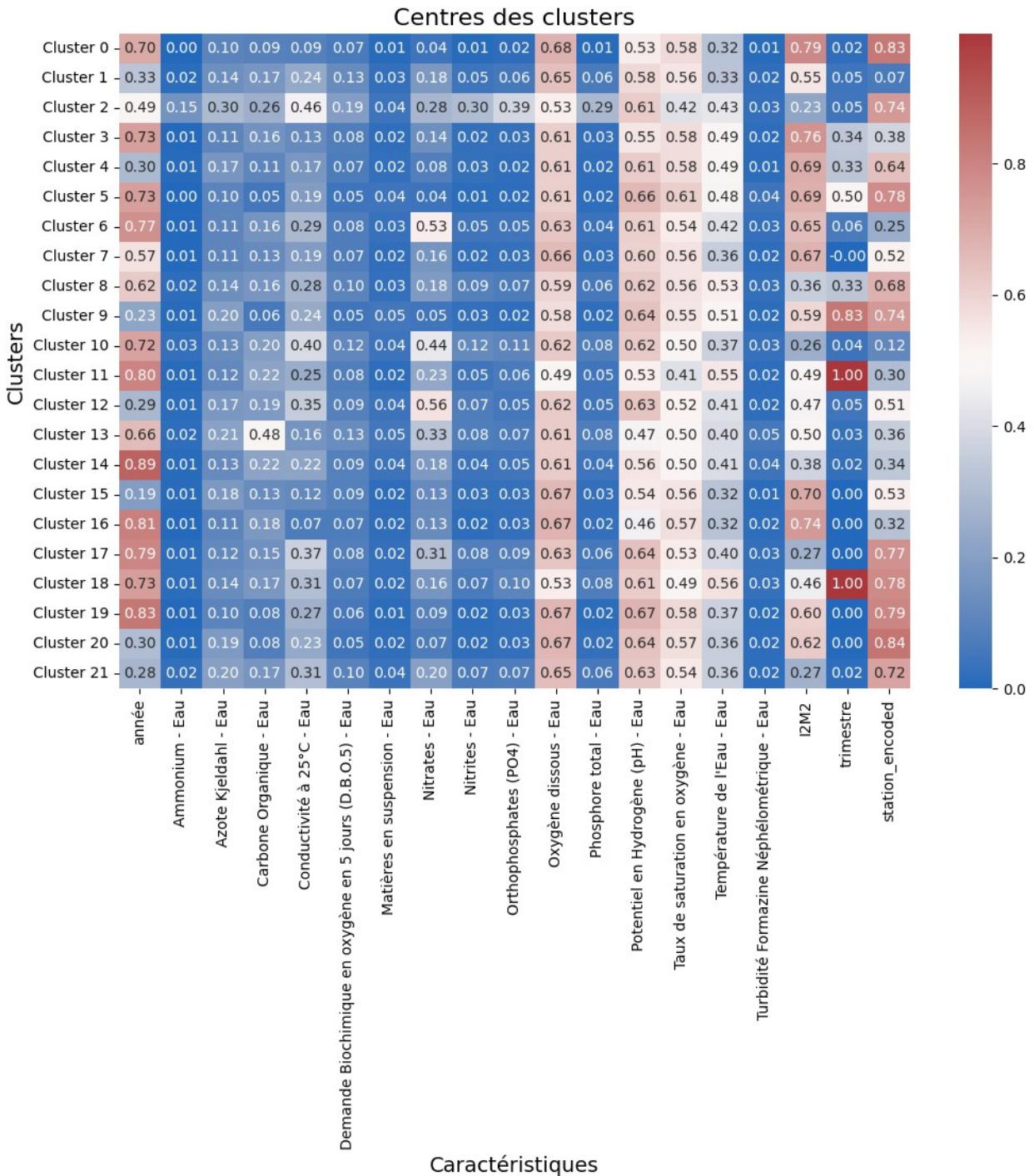
18	0.455915	9.953887e-01	0.779815
19	0.604988	1.994515e-03	0.792028
20	0.617896	4.065041e-03	0.840586
21	0.267157	1.593733e-02	0.718736
Influence des caractéristiques :			
	trimestre		0.329609
	station_encoded		0.243447
	année		0.231295
	I2M2		0.177835
	Nitrates - Eau		0.148443
	Conductivité à 25°C - Eau		0.100638
	Carbone Organique - Eau		0.089055
	Température de l'Eau - Eau		0.077369
	Orthophosphates (P04) - Eau		0.076899
	Nitrites - Eau		0.062040
	Phosphore total - Eau		0.057545
	Potentiel en Hydrogène (pH) - Eau		0.056898
	Oxygène dissous - Eau		0.050503
	Taux de saturation en oxygène - Eau		0.049415
	Azote Kjeldahl - Eau		0.048161
	Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau		0.033376
	Ammonium - Eau		0.030813
	Matières en suspension - Eau		0.012555
	Turbidité Formazine Néphélométrique - Eau		0.011044
	dtype: float64		

Après les paramètres de spatio-temporalité, les caractéristiques les plus influentes sont :

- I2M2
- Nitrates
- Conductivité à 25°C
- Carbone Organique
- Température de l'eau

Ces paramètres suggèrent des indicateurs clés de l'état de l'eau, permettant ainsi de détecter les hydroécorégions.

```
# Matrice des centres des clusters
plt.figure(figsize=(12, 8))
sns.heatmap(cluster_centers_6, annot=True, fmt=".2f", cmap="vlag",
            xticklabels=df_clustering_6.columns, yticklabels=[f'Cluster {i}' for i
            in range(len(cluster_centers_6))])
plt.title("Centres des clusters", fontsize=16)
plt.xlabel("Caractéristiques", fontsize=14)
plt.ylabel("Clusters", fontsize=14)
plt.show()
```



```
df_pc_median_bio_median_6_month_with_clusters =
df_pc_median_bio_median_6_month_c.copy()
df_pc_median_bio_median_6_month_with_clusters['cluster'] =
kmeans.labels_

# Calcul du score de silhouette
score_6 = silhouette_score(normalized_data_6,
```

```

df_pc_median_bio_median_6_month_with_clusters['cluster'])
print(f"Silhouette score: {score_6}")

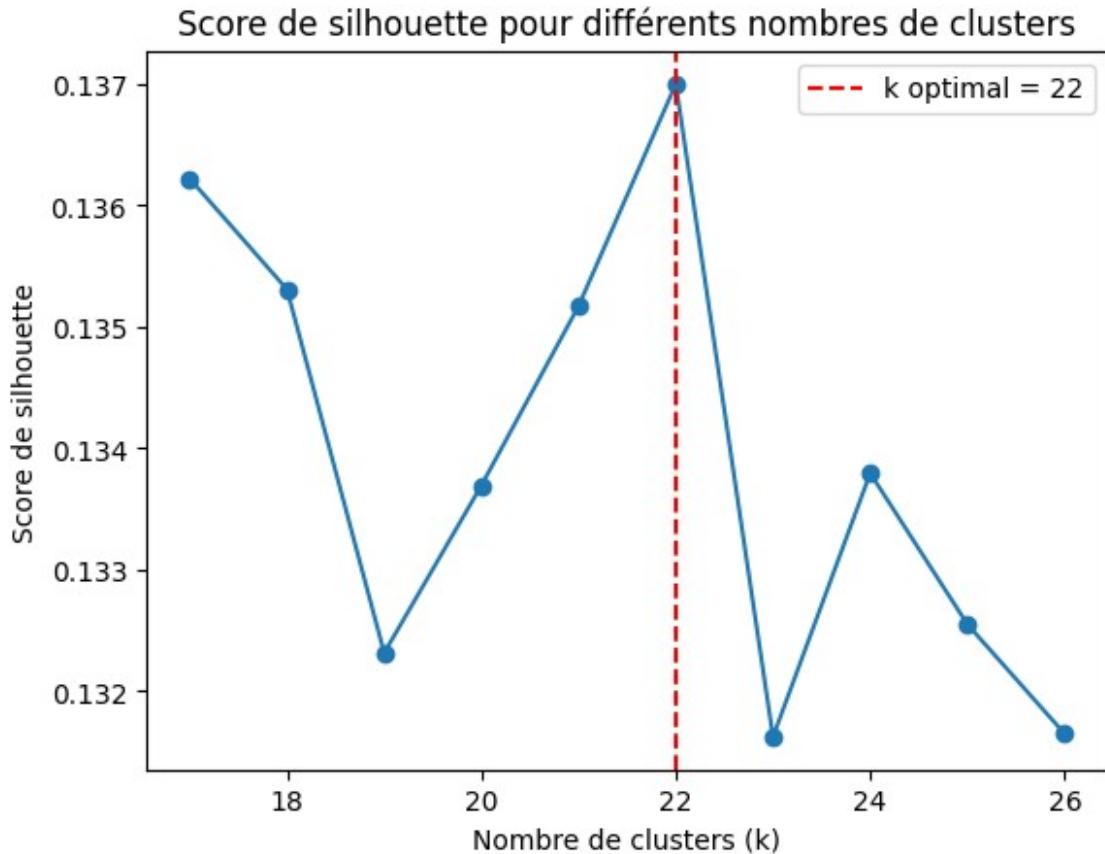
Silhouette score: 0.1370000820490834

# Recherche du meilleur nombre de clusters selon le score de
silhouette
silhouette_scores_6 = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    cluster_labels = kmeans.fit_predict(normalized_data_6)
    score = silhouette_score(normalized_data_6, cluster_labels)
    silhouette_scores_6.append(score)
best_k_6 =
k_values[silhouette_scores_6.index(max(silhouette_scores_6))]
print(f"Le meilleur nombre de clusters est : {best_k_6}")

Le meilleur nombre de clusters est : 22

plt.plot(k_values, silhouette_scores_6, marker='o')
plt.title("Score de silhouette pour différents nombres de clusters")
plt.xlabel("Nombre de clusters (k)")
plt.ylabel("Score de silhouette")
plt.axvline(x=best_k_6, color='r', linestyle='--', label=f"k optimal = {best_k_6}")
plt.legend()
plt.show()

```



Cette fois ci, le score de silhouette est exactement le nombre d'hydroécorégions.

```
cluster_distribution =
df_pc_median_bio_median_6_month_with_clusters.groupby('station')
['cluster'].value_counts(normalize=True).unstack().fillna(0) * 100
print(cluster_distribution)
stations_100_percent =
cluster_distribution[cluster_distribution.eq(100.0).sum(axis=1) == 1]
print("Stations à 100% dans un seul cluster : ", end=' ')
print(stations_100_percent.shape[0])
print("Pourcentage de stations 100% dans le même cluster : ",
round(stations_100_percent.shape[0] /
df_pc_median_bio_median_6_month_with_clusters['station'].nunique() * 100, 2), "%")

cluster      0          1          2          3          4          5
6    7    \
station

1000477  0.0  0.000000  0.000000  0.000000  0.0  0.000000
14.285714  0.0
1000602  0.0  0.000000  0.000000  0.000000  0.0  0.000000
0.000000  0.0
```

1000605	0.0	0.000000	0.000000	16.666667	0.0	0.000000
50.000000	0.0					
1001122	0.0	28.571429	0.000000	0.000000	0.0	0.000000
14.285714	0.0					
1001131	0.0	28.571429	0.000000	0.000000	0.0	0.000000
0.000000	0.0					
...
...	...					
6999125	0.0	0.000000	16.666667	0.000000	0.0	0.000000
0.000000	0.0					
6999137	0.0	0.000000	0.000000	0.000000	0.0	16.666667
0.000000	0.0					
6999153	0.0	0.000000	0.000000	0.000000	0.0	0.000000
0.000000	0.0					
6999176	0.0	0.000000	0.000000	0.000000	0.0	0.000000
0.000000	0.0					
6999178	0.0	0.000000	0.000000	0.000000	0.0	0.000000
0.000000	0.0					
cluster	8	9	...	12	13	14
16	17	\				15
station			...			
1000477	0.000000	0.0	...	0.0	0.000000	0.000000
0.000000	0.0					
1000602	0.000000	0.0	...	0.0	0.000000	0.000000
0.000000	0.0					
1000605	0.000000	0.0	...	0.0	0.000000	0.000000
0.000000	0.0					
1001122	0.000000	0.0	...	0.0	0.000000	14.285714
14.285714	0.0					
1001131	0.000000	0.0	...	0.0	14.285714	14.285714
0.000000	0.0					
...
...	...					
6999125	0.000000	0.0	...	0.0	0.000000	0.000000
0.000000	25.0					
6999137	16.666667	0.0	...	0.0	0.000000	0.000000
0.000000	0.0					
6999153	25.000000	0.0	...	0.0	0.000000	0.000000
0.000000	75.0					
6999176	50.000000	0.0	...	0.0	0.000000	0.000000
0.000000	50.0					
6999178	50.000000	0.0	...	0.0	0.000000	0.000000
0.000000	0.0					
cluster	18		19	20	21	
station						
1000477	0.000000	0.000000	0.0	0.0		
1000602	0.000000	0.000000	0.0	0.0		

```

1000605    0.000000    0.000000    0.0    0.0
1001122    0.000000    0.000000    0.0    0.0
1001131    0.000000    0.000000    0.0    0.0
...
6999125    16.666667   16.666667   0.0    25.0
6999137    0.000000    66.666667   0.0    0.0
6999153    0.000000    0.000000    0.0    0.0
6999176    0.000000    0.000000    0.0    0.0
6999178    0.000000    50.000000   0.0    0.0

[2853 rows x 22 columns]
Stations à 100% dans un seul cluster : 692
Pourcentage de stations 100% dans le même cluster : 24.26 %

df_pc_median_bio_median_6_month_with_clusters_with_coord =
df_pc_median_bio_median_6_month_with_coords.copy()
df_pc_median_bio_median_6_month_with_clusters_with_coord['cluster'] =
kmeans.labels_
# Attribution du cluster dominant à chaque station
station_cluster_counts =
df_pc_median_bio_median_6_month_with_clusters_with_coord.groupby(['sta
tion', 'cluster']).size().reset_index(name='count')
dominant_cluster_per_station =
station_cluster_counts.loc[station_cluster_counts.groupby('station')
['count'].idxmax()]
df_pc_median_bio_median_6_month_with_clusters_with_coord =
df_pc_median_bio_median_6_month_with_clusters_with_coord.merge(dominan
t_cluster_per_station[['station', 'cluster']], on='station',
how='left', suffixes=('','_dominant'))
df_pc_median_bio_median_6_month_with_clusters_with_coord.rename(column
s={'cluster_dominant': 'cluster_dominant_station'}, inplace=True)
df_analyse_6 =
df_pc_median_bio_median_6_month_with_clusters_with_coord[['station',
'année', 'CoordXStationMesureEauxSurface',
'CoordYStationMesureEauxSurface', 'trimestre',
'cluster_dominant_station']].copy()

carto_correct_i2m2_6 = gpd.GeoDataFrame(df_analyse_6, crs=crs_lambert,
geometry=gpd.GeoSeries(df_analyse_6.agg(lambda x:
geom.Point(x.loc['CoordXStationMesureEauxSurface'],
x.loc['CoordYStationMesureEauxSurface']), axis=1)))
HER_stations_correct_6 =
carto_correct_i2m2_6.sjoin(df_hydroregions.to_crs(crs_lambert),
predicate='within').to_crs(crs_lambert)
df_analyse_etude_6 = HER_stations_correct_6.drop(columns=['geometry',
'index_right', 'gid'])
df_analyse_etude_6.rename(columns={'CdHER1': 'hydroecoregion'},
inplace=True)
# hydroécorégion dominante par cluster

```

```

dominant_class_per_cluster_6 =
df_analyse_etude_6.groupby('cluster_dominant_station')
['hydroecoregion'].agg(lambda x: x.mode()[0]).reset_index()
dominant_class_per_cluster_6.columns = ['cluster_dominant_station',
'dominant_hydroecoregion_cluster']
df_analyse_etude_6 =
df_analyse_etude_6.merge(dominant_class_per_cluster_6,
on='cluster_dominant_station', how='left')
df_analyse_etude_6.drop_duplicates(subset='station', inplace=True)

# association cluster - hydroécorégion
clusters_hydroregions_6 =
df_analyse_etude_6.groupby('cluster_dominant_station')
['hydroecoregion'].apply(lambda x: x.mode()[0]).reset_index()
clusters_hydroregions_6.columns = ['cluster_dominant_station',
'dominant_hydroecoregion_cluster']
clusters_hydroregions_6['dominant_hydroecoregion_cluster'] =
clusters_hydroregions_6['dominant_hydroecoregion_cluster'].astype(str)
df_hydroregions['CdHER1'] = df_hydroregions['CdHER1'].astype(str)
clusters_hydroregions_with_names_6 =
clusters_hydroregions_6.merge(df_hydroregions[['CdHER1', 'NomHER1']],
left_on='dominant_hydroecoregion_cluster', right_on='CdHER1',
how='left')
clusters_hydroregions_with_names_6 =
clusters_hydroregions_with_names_6.rename(columns={'NomHER1':
'nom_dominant_hydroecoregion_cluster'})
print(clusters_hydroregions_with_names_6[['cluster_dominant_station',
'dominant_hydroecoregion_cluster',
'nom_dominant_hydroecoregion_cluster']])

```

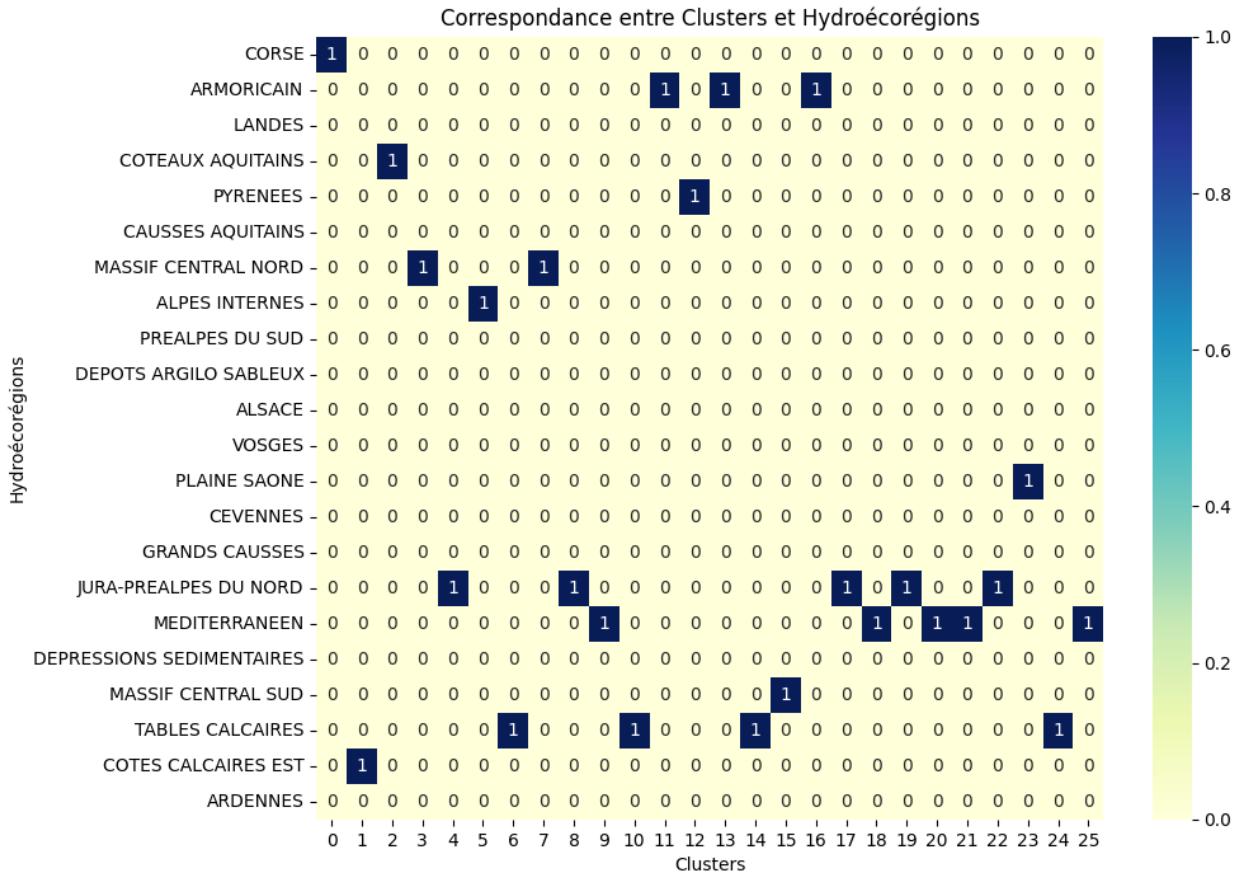
	cluster_dominant_station	dominant_hydroecoregion_cluster	\
0	0		16
1	1		10
2	2		14
3	3		21
4	4		5
5	5		2
6	6		9
7	7		21
8	8		5
9	9		6
10	10		9
11	11		12
12	12		1
13	13		12
14	14		9
15	15		3
16	16		12
17	17		5
18	18		6

19	19	5
20	20	6
21	21	6
22	22	5
23	23	15
24	24	9
25	25	6

```

  nom_dominant_hydroecoregion_cluster
0                  CORSE
1          COTES CALCAIRES EST
2          COTEAUX AQUITAINS
3      MASSIF CENTRAL NORD
4 JURA-PREALPES DU NORD
5          ALPES INTERNES
6          TABLES CALCAIRES
7      MASSIF CENTRAL NORD
8 JURA-PREALPES DU NORD
9          MEDITERRANEEN
10         TABLES CALCAIRES
11          ARMORICAIN
12          PYRENEES
13          ARMORICAIN
14         TABLES CALCAIRES
15      MASSIF CENTRAL SUD
16          ARMORICAIN
17 JURA-PREALPES DU NORD
18          MEDITERRANEEN
19 JURA-PREALPES DU NORD
20          MEDITERRANEEN
21          MEDITERRANEEN
22 JURA-PREALPES DU NORD
23          PLAINE SAONE
24         TABLES CALCAIRES
25          MEDITERRANEEN

all_hydroecoregions = df_hydroregions['NomHER1'].unique()
cross_tab_6 =
pd.crosstab(clusters_hydroregions_with_names_6['nom_dominant_hydroecoregion_cluster'],
clusters_hydroregions_with_names_6['cluster_dominant_station']).reindex(index=all_hydroecoregions, fill_value=0)
plt.figure(figsize=(10, 8))
sns.heatmap(cross_tab_6, annot=True, fmt='d', cmap='YlGnBu',
cbar=True)
plt.title("Correspondance entre Clusters et Hydroécorégions")
plt.xlabel("Clusters")
plt.ylabel("Hydroécorégions")
plt.show()
```



Cette fois ci 12 hydroécorégions sont indentifiés.

Comme pour le clustering avec décalage temporel d'1 mois, on observe des hydroécorégions associés à plusieurs clusters, et les mêmes que pou le précédent clustering, notamment l'hydroécorégion méditerranéen.

```
df_analyse_etude_6['correct_classification'] =
df_analyse_etude_6['hydroecoregion'] ==
df_analyse_etude_6['dominant_hydroecoregion_cluster']
num_correctly_classified_6 =
df_analyse_etude_6['correct_classification'].sum()
total_stations_6 = df_analyse_etude_6.shape[0]
accuracy_6 = num_correctly_classified_6 / total_stations_6
print(f"Nombre de stations bien classées:
{num_correctly_classified_6}/{total_stations_6}")
print(f"Taux de classification correcte: {accuracy_6 * 100:.2f}%")
```

Nombre de stations bien classées: 1071/2836
Taux de classification correcte: 37.76%

On a 3% de stations mieux classés qu'avec le lag d'1 mois.

```

# Représentation des stations bien classés
correct_stations_6 =
df_analyse_etude_6[df_analyse_etude_6['correct_classification']]
carto_correct_i2m2_6 = gpd.GeoDataFrame(correct_stations_6,
crs=crs_lambert,

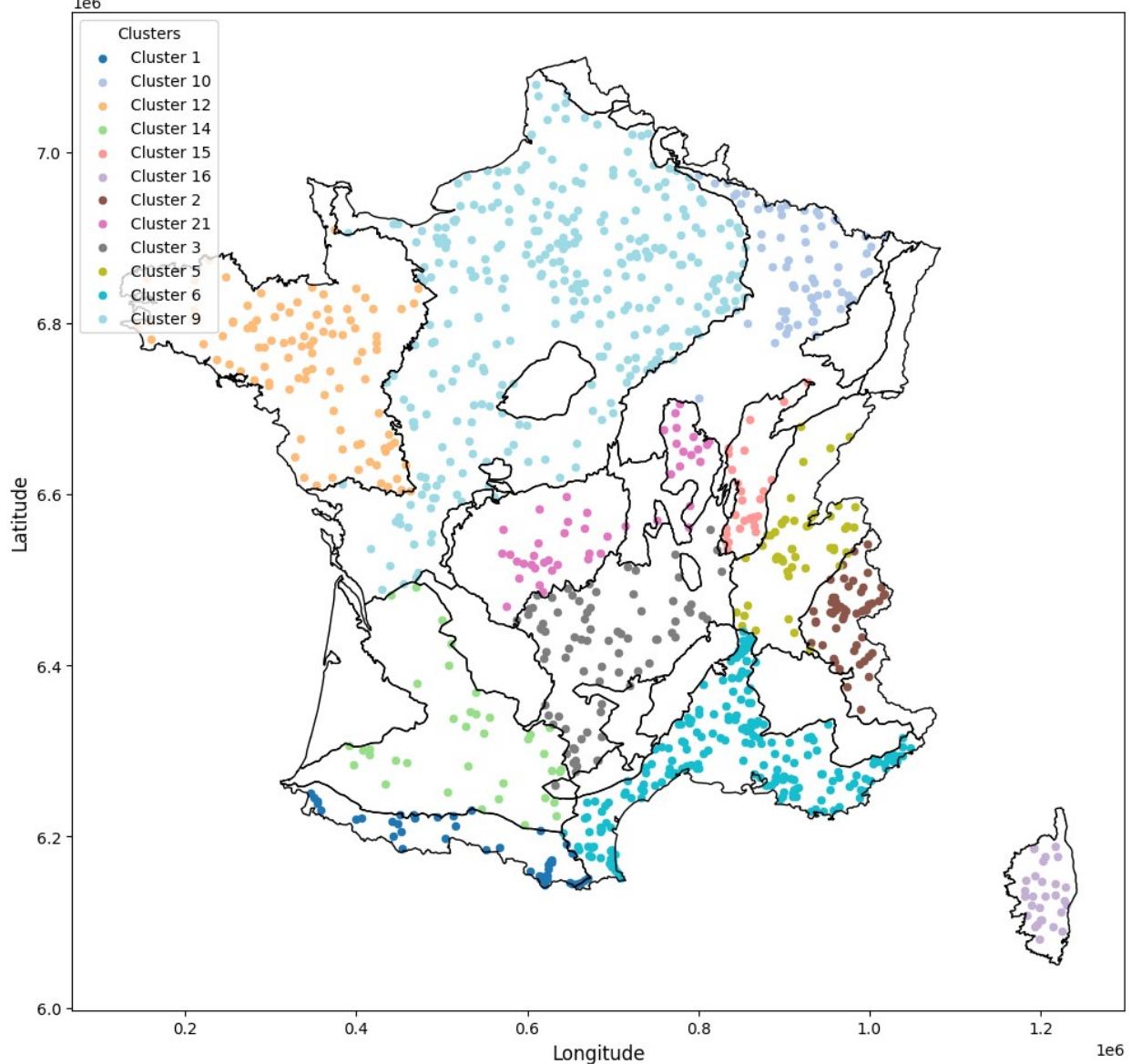
geometry=gpd.GeoSeries(correct_stations_6.apply(lambda x:
geom.Point(x[x_col], x[y_col]), axis=1)))
HER_lambert = df_hydroregions.to_crs(crs_lambert)
unique_clusters_correct_6 = carto_correct_i2m2_6[cluster_col].unique()
unique_clusters_correct_6 = sorted(unique_clusters_correct_6)
cmap = cm.get_cmap('tab20', len(unique_clusters_correct_6))
cluster_colors_correct_6 = {cluster: mcolors.to_hex(cmap(i)) for i,
cluster in enumerate(unique_clusters_correct_6)}
fig, ax = plt.subplots(1, 1, figsize=(12, 10))
HER_lambert.boundary.plot(ax=ax, color='black', linewidth=1)
for cluster, color in cluster_colors_correct_6.items():
    cluster_data =
carto_correct_i2m2_6[carto_correct_i2m2_6[cluster_col] == cluster]
    cluster_data.plot(ax=ax, color=color, markersize=20,
label=f"Cluster {cluster}")
ax.legend(loc='upper left', fontsize='medium', title='Clusters')
plt.title('Stations correctement classées par cluster avec contours
des Hydroécorégions', fontsize=16)
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.tight_layout()
plt.show()

/var/folders/p1/4yqk4cyx3058m3j04rtkr56m0000gn/T/
ipykernel_53919/3498149462.py:8: MatplotlibDeprecationWarning:

```

The `get_cmap` function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use ```matplotlib.colormaps[name]``` or ```matplotlib.colormaps.get_cmap()``` or ```pyplot.get_cmap()``` instead.

Stations correctement classées par cluster avec contours des Hydroécorégions



On observe que dans les deux clusterings, certaines régions sont plus facilement identifiables, notamment à l'ouest, dans la région de Lyon, autour de la Méditerranée, et dans le nord de la Lorraine. La Corse n'est détectée qu'avec le lag de 6 mois, tout comme la région au sud-ouest.

Les zones couvertes avec le lag de 6 mois montrent une amélioration de la précision dans la détection des hydroécorégions. En effet, le lag de 6 mois semble plus pertinent que celui de 1 mois pour étudier l'impact de l'I2M2 dans la caractérisation des hydroécorégions. Nous espérons également que cette amélioration se reflète dans la régression qui sera réalisée par la suite, en posant l'hypothèse que 6 mois offre une meilleure base pour l'analyse.

Régression : prédiction de l2M2

Dans cette section, nous explorons la possibilité de prédire l'état biologique de l'eau, représenté par l'indice l2M2, à partir des paramètres physico-chimiques, des informations spatiales (hydroécorégions) et de la dimension temporelle (saisons et décalages temporels).

En particulier, nous utilisons un lag de 6 mois pour les paramètres physico-chimiques, car les résultats du clustering précédemment réalisés avec cette configuration se sont révélés légèrement meilleurs. Cela nous conduit à supposer qu'un décalage temporel de 6 mois reflète plus fidèlement une relation sous-jacente entre les propriétés physico-chimiques et biologiques de l'eau qu'un lag d'1 mois.

Ainsi, en réutilisant les données préalablement préparées et enrichies avec l'identifiant des hydroécorégions, nous construisons un modèle de régression pour évaluer dans quelle mesure ces variables permettent de prédire efficacement l2M2. Cette démarche vise à approfondir notre compréhension des interactions entre les propriétés physico-chimiques et biologiques, tout en testant la pertinence de ces paramètres comme facteurs explicatifs de l'état des écosystèmes aquatiques.

```
df_pc_median_bio_median_6_month_with_coords  
df_for_regression = df_pc_median_bio_median_6_month_with_coords.copy()
```

```
# Pour rappel :  
df_for_regression
```

	station	année	Ammonium - Eau	Azote Kjeldahl - Eau	\
0	5001800	2007	0.040	1.0	
1	5001800	2008	0.040	1.0	
2	5001800	2009	0.060	1.0	
3	5005350	2007	0.120	1.0	
4	5005350	2008	0.040	1.0	
...	
21238	6453450	2022	0.010	0.5	
21239	6446401	2021	0.025	0.5	
21240	6446401	2022	0.020	0.5	
21241	6446330	2021	0.020	0.5	
21242	6446330	2022	0.010	0.5	

	Carbone Organique - Eau	Conductivité à 25°C - Eau	\
0	6.60	895.0	
1	5.50	803.5	
2	4.60	833.5	
3	1.65	593.0	
4	1.50	632.5	
...	
21238	2.25	456.0	
21239	1.50	503.0	
21240	2.50	463.0	
21241	1.30	529.5	
21242	1.60	541.0	

Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \

0	0.50
1	0.65
2	0.50
3	0.90
4	1.60
...	...
21238	0.60
21239	0.55
21240	1.40
21241	0.60
21242	0.90

Matières en suspension - Eau Nitrates - Eau Nitrites -

Eau	...	\			
0		11.00	45.60		
0.070	...				
1		7.50	39.25		
0.070	...				
2		10.00	41.20		
0.060	...				
3		15.00	45.40		
0.135	...				
4		21.00	44.40		
0.050	...				
...	
21238		29.50	10.40		
0.010	...				
21239		9.65	16.50		
0.030	...				
21240		24.00	12.00		
0.010	...				
21241		13.60	11.25		
0.040	...				
21242		9.10	9.20		
0.010	...				

Oxygène dissous - Eau Phosphore total - Eau \

0	11.05	0.0500
1	9.15	0.0500
2	10.70	0.0600
3	8.30	0.1600
4	10.35	0.1300
...
21238	11.35	0.0545
21239	10.85	0.0360
21240	11.00	0.0670
21241	11.05	0.0335

21242	11.10	0.0300
Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène - Eau \		
0	7.75	
100.5		
1	7.85	
90.0		
2	7.95	
91.5		
3	7.90	
80.0		
4	8.00	
95.0		
...	...	
...		
21238	7.95	
101.5		
21239	7.95	
97.5		
21240	7.90	
98.0		
21241	8.20	
99.5		
21242	8.20	
100.0		
Température de l'Eau - Eau Turbidité Formazine Néphéломétrique		
- Eau \		
0	11.35	
5.00		
1	12.85	
5.00		
2	8.25	
5.00		
3	11.80	
5.00		
4	10.70	
5.00		
...	...	
...		
21238	9.00	
20.50		
21239	9.15	
10.35		
21240	9.30	
36.00		
21241	8.95	
12.45		

```

21242          9.30
8.60

        I2M2  CoordXStationMesureEauxSurface
CoordYStationMesureEauxSurface \
0      0.3004            399856.0
6531980.0
1      0.3848            399856.0
6531980.0
2      0.5756            399856.0
6531980.0
3      0.4851            450151.0
6572920.0
4      0.3488            450151.0
6572920.0
...
...
21238  0.7330            983481.0
6700828.0
21239  0.5530            965503.0
6707708.0
21240  0.5180            965503.0
6707708.0
21241  0.1790            971794.0
6709348.0
21242  0.2380            971794.0
6709348.0

        trimestre
0              1
1              2
2              1
3              2
4              1
...
...
21238          1
21239          1
21240          1
21241          1
21242          1

[21243 rows x 21 columns]

# Ajouter le Code HER1
test= df_for_regression.copy()
carto_her = gpd.GeoDataFrame(test, crs=crs_lambert,
geometry=gpd.GeoSeries(test.agg(lambda x:
geom.Point(x.loc['CoordXStationMesureEauxSurface'],
x.loc['CoordYStationMesureEauxSurface']), axis=1)))
HER_stations_her =

```

```

carto_her.sjoin(df_hydroregions.to_crs(crs_lambert),
predicate='within').to_crs(crs_lambert)
test = HER_stations_her.drop(columns=['geometry', 'index_right',
'gid', 'NomHER1'])
test.rename(columns={'CdHER1': 'hydroecoregion'}, inplace=True)
test.head(5)

```

	station	année	Ammonium - Eau	Azote Kjeldahl - Eau	\
0	5001800	2007	0.04		1.0
1	5001800	2008	0.04		1.0
2	5001800	2009	0.06		1.0
3	5005350	2007	0.12		1.0
4	5005350	2008	0.04		1.0

	Carbone Organique - Eau	Conductivité à 25°C - Eau	\
0	6.60	895.0	
1	5.50	803.5	
2	4.60	833.5	
3	1.65	593.0	
4	1.50	632.5	

	Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	\
0	0.50	
1	0.65	
2	0.50	
3	0.90	
4	1.60	

	Matières en suspension - Eau	Nitrates - Eau	Nitrites - Eau	\
0	11.0	45.60	0.070	...
1	7.5	39.25	0.070	...
2	10.0	41.20	0.060	...
3	15.0	45.40	0.135	...
4	21.0	44.40	0.050	...

	Phosphore total - Eau	Potentiel en Hydrogène (pH) - Eau	\
0	0.05	7.75	
1	0.05	7.85	
2	0.06	7.95	
3	0.16	7.90	
4	0.13	8.00	

	Taux de saturation en oxygène - Eau	Température de l'Eau - Eau	\
0	100.5	11.35	

```
1          90.0      12.85
2          91.5      8.25
3          80.0      11.80
4          95.0      10.70
```

```
Turbidité Formazine Néphélométrique - Eau    I2M2  \
0          5.0  0.3004
1          5.0  0.3848
2          5.0  0.5756
3          5.0  0.4851
4          5.0  0.3488
```

```
CoordXStationMesureEauxSurface  CoordYStationMesureEauxSurface
trimestre \
0          399856.0      6531980.0
1          399856.0      6531980.0
2          399856.0      6531980.0
3          450151.0      6572920.0
2          450151.0      6572920.0
4          450151.0      6572920.0
1
```

```
hydroecoregion
0          9
1          9
2          9
3          9
4          9
```

```
[5 rows x 22 columns]
```

```
# drop les coordonnées
df_for_regression = test.copy()
df_for_regression.drop(columns=['CoordXStationMesureEauxSurface',
'CoordYStationMesureEauxSurface'], inplace=True)
df_for_regression.head(5)
```

```
station  année  Ammonium - Eau  Azote Kjeldahl - Eau  \
0  5001800  2007      0.04      1.0
1  5001800  2008      0.04      1.0
2  5001800  2009      0.06      1.0
3  5005350  2007      0.12      1.0
4  5005350  2008      0.04      1.0
```

```
Carbone Organique - Eau  Conductivité à 25°C - Eau \
0          6.60      895.0
1          5.50      803.5
```

2	4.60	833.5
3	1.65	593.0
4	1.50	632.5
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau \		
0	0.50	
1	0.65	
2	0.50	
3	0.90	
4	1.60	
Matières en suspension - Eau Nitrates - Eau Nitrites - Eau \		
0	11.0	45.60
1	7.5	39.25
2	10.0	41.20
3	15.0	45.40
4	21.0	44.40
Orthophosphates (P04) - Eau Oxygène dissous - Eau Phosphore total		
- Eau \		
0	0.050	11.05
0.05		
1	0.050	9.15
0.05		
2	0.050	10.70
0.06		
3	0.245	8.30
0.16		
4	0.170	10.35
0.13		
Potentiel en Hydrogène (pH) - Eau Taux de saturation en oxygène -		
Eau \		
0	7.75	
100.5		
1	7.85	
90.0		
2	7.95	
91.5		
3	7.90	
80.0		
4	8.00	
95.0		
Température de l'Eau - Eau Turbidité Formazine Néphéломétrique -		
Eau \		
0	11.35	
5.0		
1	12.85	
5.0		

```

2                      8.25
5.0
3                      11.80
5.0
4                      10.70
5.0

    I2M2  trimestre hydroecoregion
0  0.3004            1            9
1  0.3848            2            9
2  0.5756            1            9
3  0.4851            2            9
4  0.3488            1            9

# valeur min et max de I2M2 : vérifier que c'est bien 0 et 1
print(df_for_regression['I2M2'].min())
print(df_for_regression['I2M2'].max())

0.0
1.0

from sklearn.preprocessing import LabelEncoder, MinMaxScaler
# Encodage des stations
le = LabelEncoder()
df_for_regression['station_encoded'] =
le.fit_transform(df_for_regression['station'])
df_for_regression["hydroecoregion_encoded"] =
le.fit_transform(df_for_regression["hydroecoregion"])
df_for_regression.drop(columns=['station', 'hydroecoregion'],
inplace=True)
# Normalisation des données pour qu'elles aient toutes la même importance
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(df_for_regression)
normalized_data = pd.DataFrame(normalized_data,
columns=df_for_regression.columns)
# df_for_regression = pd.concat([df_for_regression[['station_encoded', 'hydroecoregion_encoded']], normalized_data], axis=1)

from sklearn.model_selection import train_test_split
X = normalized_data.drop(columns=['I2M2'])
y = normalized_data['I2M2']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
model = LinearRegression()
cv_scores = cross_val_score(model, X_train, y_train, cv=5,
scoring='neg_mean_squared_error')

```

```

mean_cv_score = -cv_scores.mean()
print(f"Mean CV score: {mean_cv_score}")

Mean CV score: 0.03133060488047301

```

Le fait que y soit compris entre 0 et 1 est à prendre en compte.

Un score moyen de validation croisée (Mean CV score) dans une régression représente l'erreur quadratique moyenne (MSE) qui mesure la différence entre les valeurs réelles et les valeurs prédites par le modèle. Le MSE est exprimé dans les mêmes unités que la variable cible.

Puisque la plage de y est de 0 à 1, un MSE de 0.03 signifie que, en moyenne, l'écart quadratique entre les valeurs réelles et prédites est de 0.03, ce qui est relativement faible par rapport à l'échelle totale (0 à 1).

```

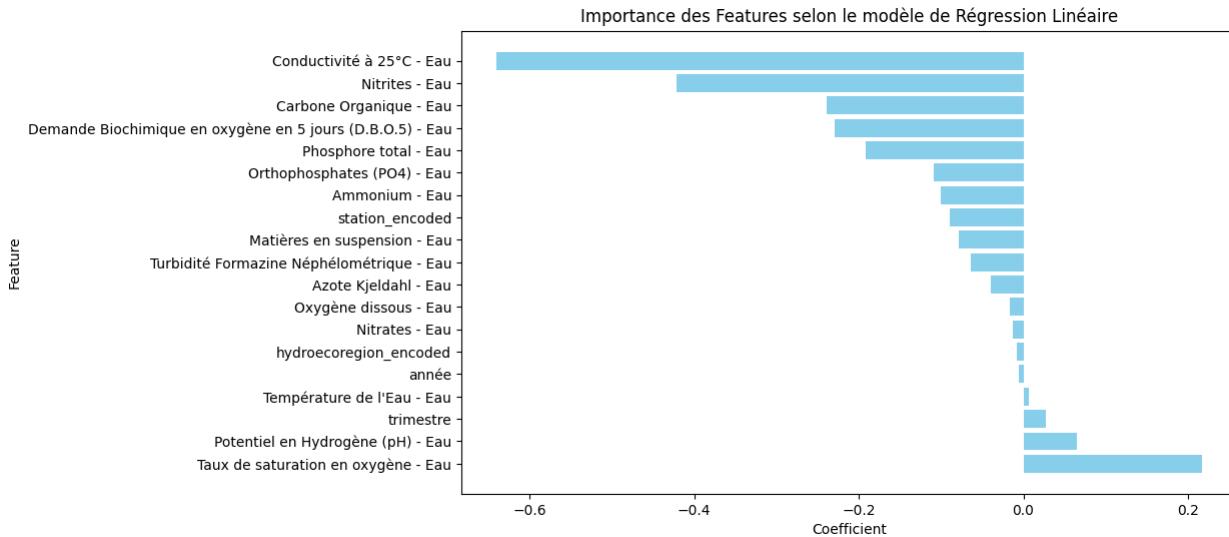
model.fit(X_train, y_train)
coefficiens = model.coef_
features = X_train.columns
coef_df = pd.DataFrame(coefficiens, index=features,
columns=['Coefficient'])
coef_df = coef_df.sort_values('Coefficient', ascending=False)
print(coef_df)

                                         Coefficient
Taux de saturation en oxygène - Eau          0.216832
Potentiel en Hydrogène (pH) - Eau           0.064725
trimestre                                     0.027391
Température de l'Eau - Eau                  0.007042
année                                         -0.005469
hydroecoregion_encoded                      -0.008531
Nitrates - Eau                                -0.013270
Oxygène dissous - Eau                         -0.015859
Azote Kjeldahl - Eau                          -0.039886
Turbidité Formazine Néphélométrique - Eau   -0.063309
Matières en suspension - Eau                 -0.078234
station_encoded                               -0.089366
Ammonium - Eau                                -0.100480
Orthophosphates (P04) - Eau                  -0.108816
Phosphore total - Eau                         -0.191791
Demande Biochimique en oxygène en 5 jours (D.B.... -0.228816
Carbone Organique - Eau                       -0.238602
Nitrites - Eau                                 -0.420908
Conductivité à 25°C - Eau                   -0.640112

# Visualisation des coefficients
plt.figure(figsize=(10, 6))
plt.barh(coef_df.index, coef_df['Coefficient'], color='skyblue')
plt.xlabel('Coefficient')
plt.ylabel('Feature')
plt.title('Importance des Features selon le modèle de Régression')

```

```
Linéaire')  
plt.show()
```



On regarde les caractéristiques qui participent le plus à la détermination de l'indice I2M2 en regardant les valeurs absolues des différents paramètres.

```
coef_df_sorted_abs = coef_df.abs().sort_values('Coefficient',  
ascending=False)  
print(coef_df_sorted_abs)
```

	Coefficient
Conductivité à 25°C - Eau	0.640112
Nitrites - Eau	0.420908
Carbone Organique - Eau	0.238602
Demande Biochimique en oxygène en 5 jours (D.B.O.5) - Eau	0.228816
Taux de saturation en oxygène - Eau	0.216832
Phosphore total - Eau	0.191791
Orthophosphates (PO4) - Eau	0.108816
Ammonium - Eau	0.100480
station_encoded	0.089366
Matières en suspension - Eau	0.078234
Potentiel en Hydrogène (pH) - Eau	0.064725
Turbidité Formazine Néphélométrique - Eau	0.063309
Azote Kjeldahl - Eau	0.039886
trimestre	0.027391
Oxygène dissous - Eau	0.015859
Nitrates - Eau	0.013270
hydroecoregion_encoded	0.008531
Température de l'Eau - Eau	0.007042
année	0.005469

Résultats

D'après les coefficients les plus éloignés de zéro, les six caractéristiques les plus influentes pour la prédiction de l2M2 sont, dans l'ordre :

- Conductivité à 25°C - Eau
 - Nitrites - Eau
 - Carbone Organique - Eau
 - Demande Biochimique en oxygène en 5 jours
 - Taux de saturation en oxygène - Eau
 - Phosphore total - Eau
-

Régularisation

Nous utilisons la régression Lasso, qui applique une régularisation L1, pour identifier les caractéristiques ayant un véritable impact sur la prédiction de l2M2. Cette régularisation L1 permet de réduire à zéro les coefficients des variables les moins influentes, ce qui nous aide à déterminer quelles caractéristiques physicochimiques ont le plus d'impact sur les données hydrobiologiques.

```
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=0.005)
cv_scores = cross_val_score(lasso, X_train, y_train, cv=5,
scoring='neg_mean_squared_error')
mean_cv_score = -cv_scores.mean()

print(f"Mean CV score: {mean_cv_score}")

Mean CV score: 0.03731374537661418

lasso.fit(X_train, y_train)
lasso_coef = lasso.coef_
lasso_coef_df = pd.DataFrame(lasso_coef, index=features,
columns=['Coefficient'])
lasso_coef_df = lasso_coef_df.sort_values('Coefficient',
ascending=False)
print(lasso_coef_df)

          Coefficient
année           0.000000
Oxygène dissous - Eau      0.000000
station_encoded        -0.000000
trimestre            0.000000
Turbidité Formazine Néphélométrique - Eau     -0.000000
Température de l'Eau - Eau      -0.000000
Taux de saturation en oxygène - Eau      0.000000
Potentiel en Hydrogène (pH) - Eau      0.000000
```

```

Phosphore total - Eau           -0.000000
Orthophosphates (P04) - Eau     -0.000000
Ammonium - Eau                  -0.000000
Nitrites - Eau                  -0.000000
Matières en suspension - Eau    -0.000000
Demande Biochimique en oxygène en 5 jours (D.B....) -0.000000
Azote Kjeldahl - Eau           -0.000000
hydroecoregion_encoded         0.000000
Nitrates - Eau                  -0.001346
Carbone Organique - Eau         -0.092639
Conductivité à 25°C - Eau       -0.528447

print(f"Nombre de features gardées : {sum(lasso_coef != 0)}")

Nombre de features gardées : 3

print("Résultats : les caractéristiques les plus importantes selon le modèle Lasso")
df_lasso_coef = lasso_coef_df[lasso_coef_df['Coefficient'] != 0]
df_lasso_coef_sorted = df_lasso_coef.abs().sort_values('Coefficient', ascending=False)
print(df_lasso_coef_sorted)

Résultats : les caractéristiques les plus importantes selon le modèle Lasso
                                         Coefficient
Conductivité à 25°C - Eau          0.528447
Carbone Organique - Eau             0.092639
Nitrates - Eau                     0.001346

```

On retrouve des paramètres précédemment mentionnés.

Conclusion

Notre travail a exploré la relation entre les données physicochimiques et hydrobiologiques dans le but de mieux comprendre les interactions entre ces paramètres et d'identifier des modèles spatiaux significatifs, tels que les hydroécorégions.

L'objectif principal était de déterminer si ces régions écologiques pouvaient être détectées à partir des données physicochimiques et hydrobiologiques, et comment les différents paramètres interagissent pour influencer l'état des écosystèmes aquatiques.

Nous avons choisi de prendre en compte différents laps de temps, à savoir 1 mois et 6 mois, afin d'étudier les relations de causalité entre les paramètres physicochimiques et les indicateurs hydrobiologiques. Cette approche nous a permis d'analyser les corrélations entre les variables et de nous interroger sur l'influence persistante de certains paramètres physicochimiques, malgré les variations saisonnières, sur l'état hydrobiologique. Ainsi, nous avons exploré dans quelle mesure la temporalité impacte ces interactions et si les paramètres physicochimiques peuvent prédire ou expliquer les dynamiques des écosystèmes aquatiques sur différentes périodes.

L'application du clustering K-means a introduit une nouvelle dimension dans notre étude : celle de l'identification des hydroécorégions à partir des données. Ce processus nous a permis de tester si des regroupements naturels des stations, basés sur leurs caractéristiques physicochimiques et hydrobiologiques, étaient observables. Nous avons également comparé l'impact des deux laps de temps (1 mois vs 6 mois) pour déterminer lequel offrait de meilleurs résultats dans la détection des hydroécorégions, en fonction de la dynamique des paramètres et de l'évolution des conditions environnementales.

Après avoir identifié le laps de temps optimal, nous avons réintégré les hydroécorégions dans notre dataset, puis entraîné un modèle de régression pour prédire I2M2. Cela nous a permis de déterminer les trois paramètres les plus influents pour la prédiction de I2M2. Enfin, nous avons appliqué une approche de régularisation (Lasso) pour identifier les caractéristiques physicochimiques ayant le plus de poids dans la prédiction de I2M2, nous permettant ainsi de mieux comprendre quels paramètres sont véritablement significatifs et présentent un lien concret avec les dynamiques écologiques sur le terrain.

Notre travail s'est structuré autour de plusieurs étapes clés :

- Préparation et analyse des données physicochimiques et hydrobiologiques
- Fusion des différents jeux de données avec les traitements préliminaires nécessaires
- Préparation des données pour le clustering des stations : choix du nombre de clusters, encodage des variables catégorielles, et normalisation
- Réalisation du clustering des stations en utilisant la méthode K-Means pour différents lags temporels
- Analyse des résultats obtenus pour chaque lag et comparaison des performances pour déterminer le laps de temps le plus cohérent
- Visualisation des résultats
- Réalisation d'une régression pour prédire I2M2 à partir des données, avec l'intégration des hydroécorégions, et application d'une régularisation afin d'identifier les variables physicochimiques ayant un impact réel sur la prédiction de I2M2

En résumé :

- Nous avons pu identifier les paramètres physicochimiques et hydrobiologiques qui permettent de différencier les différentes régions au cours du temps
- Nous avons constaté qu'un lag de 6 mois est plus pertinent qu'un lag de 1 mois, ce qui suggère que l'impact des données physicochimiques sur l'état hydrobiologique est plus prononcé à long terme
- Nous avons déterminé les paramètres physicochimiques les plus influents dans l'explication des dynamiques hydrobiologiques.

Résultats

Les résultats du clustering des stations ont montré que, bien que prometteurs, seulement un tiers des stations ont été correctement classées dans leurs hydroécorégions respectives. Cela suggère que certaines hydroécorégions partagent des caractéristiques physicochimiques et hydrobiologiques trop similaires, rendant leur distinction difficile. En effet, les résultats obtenus avec les deux approches de clustering (1 mois et 6 mois) étaient relativement proches. Toutefois, l'approche avec un lag de 6 mois a permis de classer légèrement plus de stations correctement, avec une amélioration de 3 % par rapport à celle avec un lag de 1 mois. De plus, l'indice de

silhouette a révélé que, pour un lag de 6 mois, 22 clusters ont été détectés contre 26 pour un lag de 1 mois.

En ce qui concerne les paramètres influençant le clustering des hydroécorégions, nous avons observé que, quel que soit le lag (1 mois ou 6 mois), les paramètres les plus discriminants sont :

- I2M2
- Nitrates
- Conductivité à 25°C
- Carbone organique
- Température de l'eau

La majorité de ces paramètres se sont distingués dans les centres des clusters, ce qui indique qu'ils sont les principaux facteurs influençant la formation des clusters. En particulier, I2M2, les nitrates, le carbone organique et la conductivité de l'eau apparaissent comme les indicateurs les plus significatifs pour différencier les hydroécorégions.

Concernant les résultats de la régression, les trois paramètres les plus importants pour la caractérisation de l'indice I2M2 sont la conductivité à 25°C, les nitrites et le carbone organique dans l'eau. Les résultats du modèle Lasso confirment cette tendance, avec la conductivité à 25°C, le carbone organique et les nitrates apparaissant comme les facteurs les plus significatifs.

Nous avons réutilisé le dataset avec un décalage temporel de 6 mois car nous avons observé que cette approche semble plus pertinente pour capturer la différence de temps entre les changements dans l'eau et leur impact sur l'hydrobiologie, comme expliqué précédemment.

Il est important de noter que les coefficients négatifs de la conductivité, des nitrites et du carbone organique dans la régression indiquent que ces paramètres sont associés à une diminution de l'indice I2M2. En d'autres termes, à mesure que ces paramètres augmentent, l'indice I2M2 a tendance à diminuer, ce qui pourrait suggérer que des concentrations plus élevées de ces éléments ont un impact négatif sur l'état hydrobiologique de l'eau.

Les matrices de corrélation ont également révélé des corrélations négatives entre I2M2, la conductivité à 25°C et les nitrates, renforçant ainsi l'idée que ces paramètres jouent un rôle central dans l'état de l'hydrobiologie de l'eau.

NB : À noter que nous avons également testé l'approche sans décalage temporel (comme détaillé dans ce notebook), mais les résultats se sont avérés clairement moins bons que ceux obtenus avec un décalage de 1 mois.

Limitations et biais

Pour le clustering des hydroécorégions :

- Surreprésentation des données hydrobiologiques en été et sous-représentation en dehors de cette période.
- Les données physicochimiques sont régulières, mais l'hydrobiologie est biaisée par la saisonnalité.
- Lors de la jointure des données, perte de 50 % des stations, réduisant ainsi la taille de l'échantillon en raison du faible nombre de relevés.
- Déséquilibre dans le nombre de stations par région, ce qui introduit des biais dans l'analyse.
- Suppression de certaines combinaisons de paramètres physicochimiques en raison de leur faible représentation.
- Beaucoup d'imputation des valeurs manquantes par la médiane, ce qui peut affecter la précision des résultats.
- Suppression des outliers via la médiane, en raison du nombre limité de données, pour éviter l'impact de valeurs extrêmes. La médiane a été choisie car elle est moins sensible aux valeurs aberrantes et reflète mieux la tendance centrale dans des ensembles de données hétérogènes.

Pour la régression :

- Une méthode de clustering améliorée, permettant de mieux ajuster le lag temporel entre les données physicochimiques et hydrobiologiques, pourrait potentiellement conduire à des résultats plus représentatifs. En effet, on aurait pu extraire un autre dataset (avec le lag de temps "optimal").

Cependant :

- Nous avons réussi à prédire l'I2M2 avec une erreur moyenne raisonnable en régression.
 - Les mêmes paramètres importants apparaissent régulièrement entre la régression, la régularisation et les corrélations, ce qui renforce la robustesse des résultats obtenus, et suggère que l'analyse est globalement fiable.
-

Pistes

Pour le clustering des hydroécorégions :

- Tester d'autres lags de temps pour mieux capter les variations saisonnières (1 an serait très intéressant car on pourrait capturer les impacts saisonniers tout en tenant compte du décalage temporel entre les données physicochimiques et hydrobiologiques).
- Explorer différentes méthodes d'agrégation des valeurs, comme l'utilisation de la moyenne.
- Essayer d'autres agrégations temporelles (par exemple, par mois ou par année).

Une étude plus approfondie, incluant une répartition plus homogène des données et une exploration d'autres périodes temporelles, pourrait affiner la détection des hydroécorégions et améliorer la compréhension des facteurs physicochimiques et hydrobiologiques qui influencent les écosystèmes aquatiques.

Concernant la méthode de clustering, des approches alternatives comme le K-Means avec Dynamic Time Warping (DTW) ou l'utilisation de K-Medoids pourraient être envisagées. Ces techniques pourraient permettre une meilleure gestion des séries temporelles et une meilleure distinction des groupes d'hydroécorégions, en tenant compte de la nature dynamique des données.

Ces différentes perspectives ouvrent la voie à une meilleure compréhension des interactions complexes entre les facteurs physicochimiques et hydrobiologiques, et à une amélioration des méthodes d'analyse utilisées pour détecter et caractériser les hydroécorégions.