In [64]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importation des données :

Dans un premier temps nous importons les données concernant le temps de germinations d'un pourcentage de graines; NNous utilisions la librairie Pandas de Python pour lire notre jeu de de données.

In [65]:
```
df = pd.read_excel("C:/Users/ua/Desktop/betterave.xlsx","TMG-TXX_rep")
#df.head()
df1= df.copy()
df1.head()
```

Out[65]:

	Echantillon	Répétition	Echantillon_Repétition	TMG	Ecart-type TMG	T10	T20	T30	T40	T50	T60	T70	T80	T90
0	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5
1	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5
2	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5
3	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5
4	1301	2	1301_2	188.370787	20.270546	162.0	173.00	178.8	183.375	188.250	193.8	204.00	217.5	NaN

Pour traiter l'effet du choix d'un Banc sur la germination des graines de betteraves , alors on décide d'ajouter la variable "Banc" à notre jeu de données .

In [66]:
```
dfx = pd.read_excel("C:/Users/ua/Desktop/betterave.xlsx","Plan_semis")
dfx = dfx[['Banc']]
df =pd.concat([df, dfx], axis=1)
#dfxx = pd.merge(df ,dfx, on= 'Echantillon')
#df.head()
df.head()
```

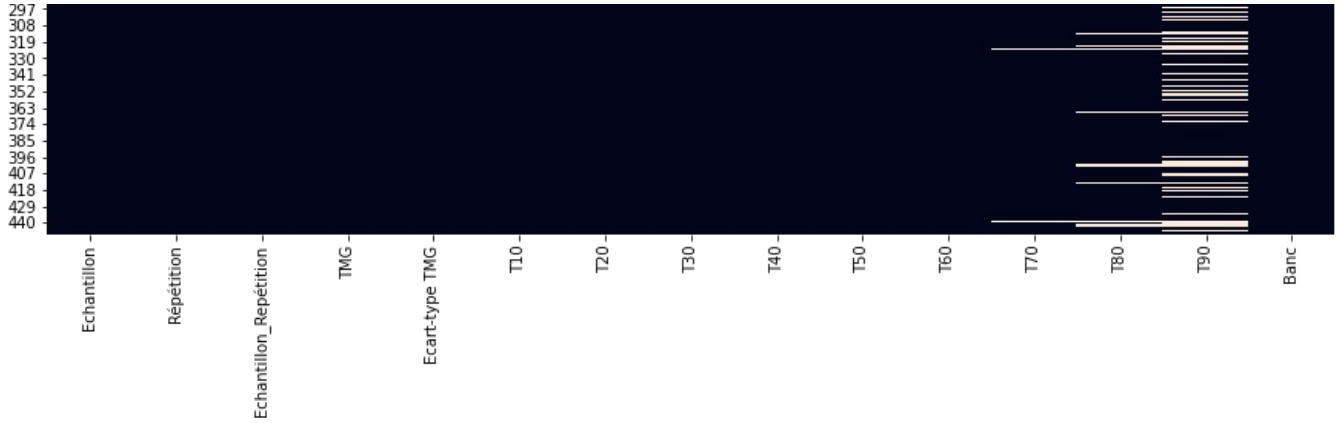
Out[66]:

	Echantillon	Répétition	Echantillon_Repétition	TMG	Ecart-type TMG	T10	T20	T30	T40	T50	T60	T70	T80	T90	Ba
0	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5	
1	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5	
2	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5	
3	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	195.6	201.75	210.0	229.5	
4	1301	2	1301_2	188.370787	20.270546	162.0	173.00	178.8	183.375	188.250	193.8	204.00	217.5	NaN	

Avant de commencer notre analyse , on peut traiter le problème des valeurs manquantes.

In [67]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.isna(),cbar=False)
```

Out[67]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ee4d0ab340>
```



In [68]:
```
(df.isna().sum()/df.shape[0]).sort_values(ascending=True)
```

Out[68]:
```
Echantillon               0.000000
Répétition                0.000000
Echantillon_Repétition    0.000000
TMG                       0.000000
Ecart-type TMG            0.000000
T10                       0.000000
T20                       0.000000
T30                       0.000000
T40                       0.000000
T50                       0.000000
T60                       0.000000
Banc                      0.000000
T70                       0.022321
T80                       0.064732
T90                       0.316964
dtype: float64
```

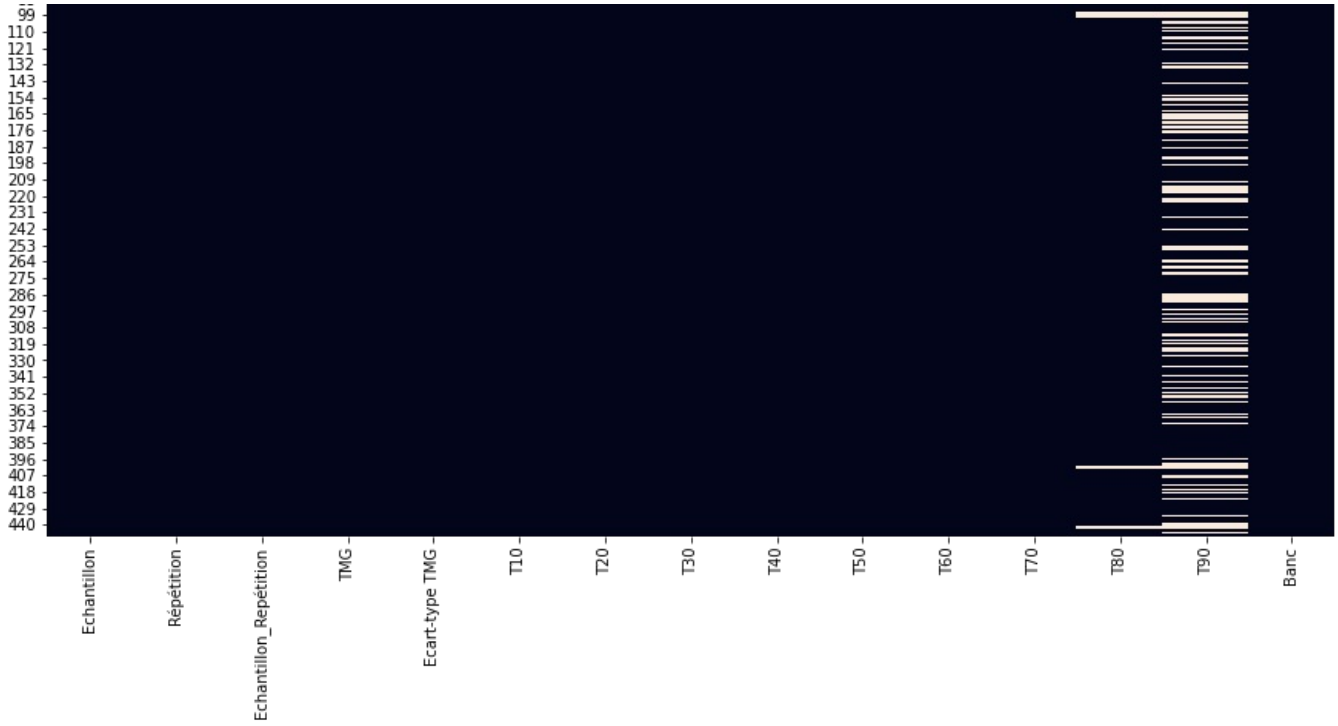
En utilisant la fonction "heatmap" de la librairie "seaborn" , on peut visualiser facilement l'emplacement des valeurs manquantes, puis on peut calculer le pourcentage des Nan pour chaque variable.On peut remarquer alors que la variables "T90" contient 31% de valeurs manquantes en ravanche les varibales "T80" et "T70" contienent respectivement que 6% et 2% de valeurs manquantes .

In [69]:
```
def Donnees_Manquantes(df,colonne):
    for i in range(df.shape[0]):
        if str(df.iloc[i,colonne]) == 'nan':
            if df.iloc[i,0] == df.iloc[i+1,0]:
    #print("cas individu suivant", i)
    #print("c'était ",df.iloc[i,13])
                df.iloc[i,colonne] =df.iloc[i+1,colonne]
    #print("c'est devenu ",df.iloc[i,colonne] )
            elif df.iloc[i,0] == df.iloc[i-1,0]:
    #print("cas individu précédent ", i)
    #print("c'était ",df.iloc[i,colonne])
                df.iloc[i,colonne] = df.iloc[i-1,colonne]
    #print("c'était ",df.iloc[i,colonne])
            else:
                pass
Donnees_Manquantes(df,11)
Donnees_Manquantes(df,12)
Donnees_Manquantes(df,11)
Donnees_Manquantes(df,12)
```

Vu que notre jeu de données contient des répétitions des individus("Echantillon") , on a alors décidé de remplacer nos valeurs NaN des variables "T80" et "T70" par des valeurs similaires du même "Echantillon" .

In [70]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.isna(),cbar=False)
```

Out[70]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ee55b9be80>
```

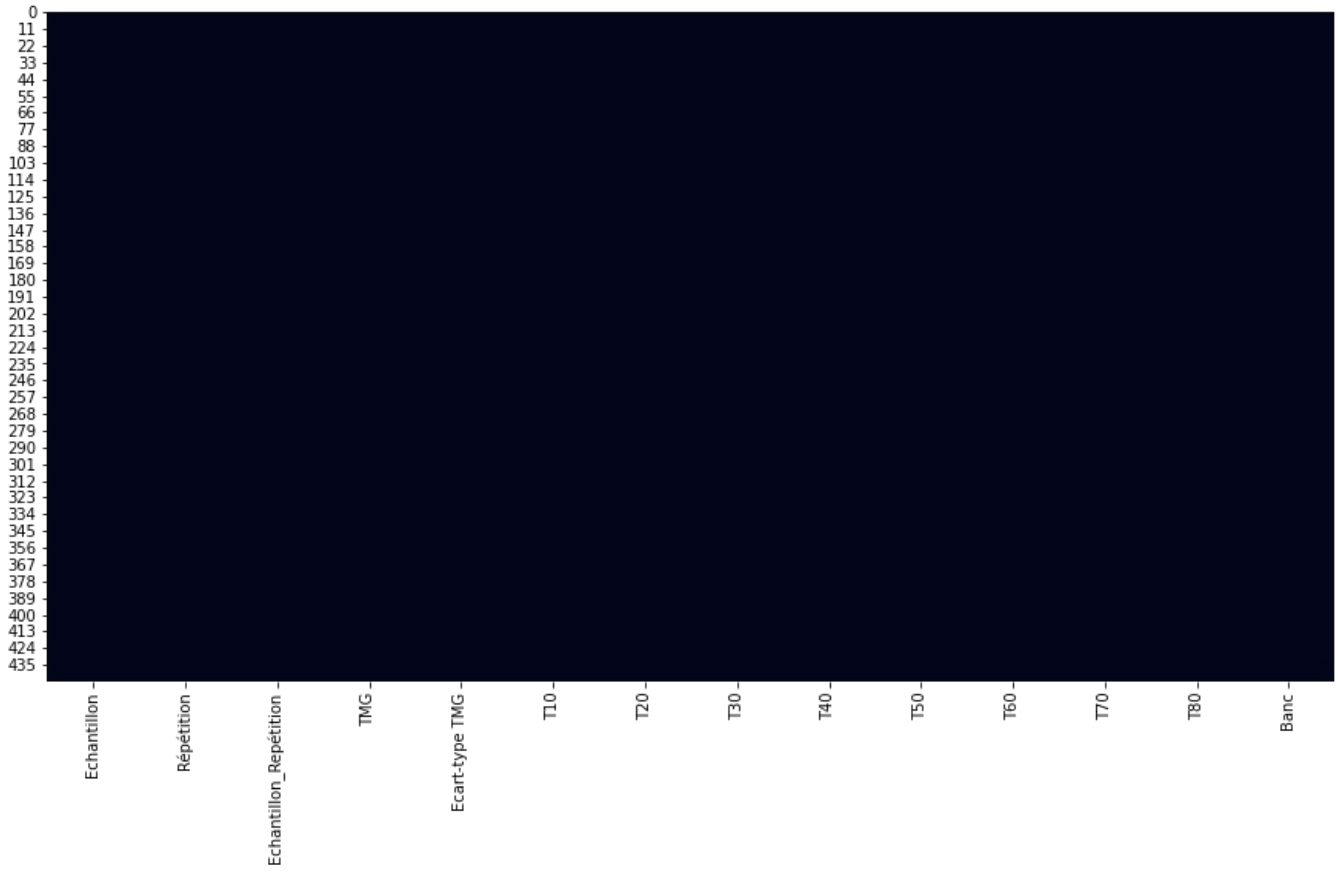


Apres une visualistation de nos données , on remarque qu'il existe toujours des valeurs manquantes pour les varibables "T80" et "T70" , donc maintenant on peut procéder à éliminer ces individus après avoir supprimer la varible "T90" .

In [71]:
```
df = df.drop(["T90"],axis =1)
```

In [72]:
```
df=df.dropna()
df
plt.figure(figsize=(15,8))
sns.heatmap(df.isna(),cbar=False)
```

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x1ee515d7fd0>



In [73]:
```
df["variation1"]= df["T20"] - df["T10"]
df["variation2"]= df["T30"] - df["T20"]
df["variation3"]= df["T40"] - df["T30"]
df["variation4"]= df["T50"] - df["T40"]
df["variation5"]= df["T60"] - df["T50"]
df["variation6"]= df["T70"] - df["T60"]
df["variation7"]= df["T80"] - df["T70"]
```

In [74]:
```
df.head()
```

Out[74]:

	Echantillon	Répétition	Echantillon_Repétition	TMG	Ecart-type TMG	T10	T20	T30	T40	T50	...	T70	T80	Banc	variati
0	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	...	201.75	210.0	4	12
1	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	...	201.75	210.0	4	12
2	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	...	201.75	210.0	4	12
3	1301	1	1301_1	185.677419	23.175212	154.5	167.25	175.2	181.200	187.125	...	201.75	210.0	4	12
4	1301	2	1301_2	188.370787	20.270546	162.0	173.00	178.8	183.375	188.250	...	204.00	217.5	4	11

5 rows × 21 columns

In [75]:
```
df.columns
```

Out[75]:
```
Index(['Echantillon', 'Répétition', 'Echantillon_Repétition', 'TMG',
       'Ecart-type TMG', 'T10', 'T20', 'T30', 'T40', 'T50', 'T60', 'T70',
       'T80', 'Banc', 'variation1', 'variation2', 'variation3', 'variation4',
       'variation5', 'variation6', 'variation7'],
      dtype='object')
```

In [76]:
```
df = df.drop(["Répétition",'Echantillon_Repétition', 'TMG',
       'Ecart-type TMG', 'T10', 'T20', 'T30', 'T40', 'T50', 'T60', 'T70',
       'T80'],axis = 1)
df.head()

#df2=df.copy()
```

Out[76]:

	Echantillon	Banc	variation1	variation2	variation3	variation4	variation5	variation6	variation7
0	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
1	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
2	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
3	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
4	1301	4	11.00	5.80	4.575	4.875	5.550	10.20	13.50

In [77]:
```
df2=df.copy()
df2.head()
```

Out[77]:

	Echantillon	Banc	variation1	variation2	variation3	variation4	variation5	variation6	variation7
0	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
1	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
2	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
3	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
4	1301	4	11.00	5.80	4.575	4.875	5.550	10.20	13.50

Effet du choix du Banc sur la germination ?

Après avoir nettoyer nos données , on peut maintenant créer des variables nommées "variation" qui consistent à calculer les variations entre deux temps consécutifs .

Visualisation des données :

In [79]:
```
#df2 = df[df["Echantillon"]== df["Echantillon"].unique()[2]]
#df2
#dfbanc4 = df[df["Banc"] == 4]
#dfbanc4.head()

df = df.groupby("Echantillon").mean()
df
```

Out[79]:

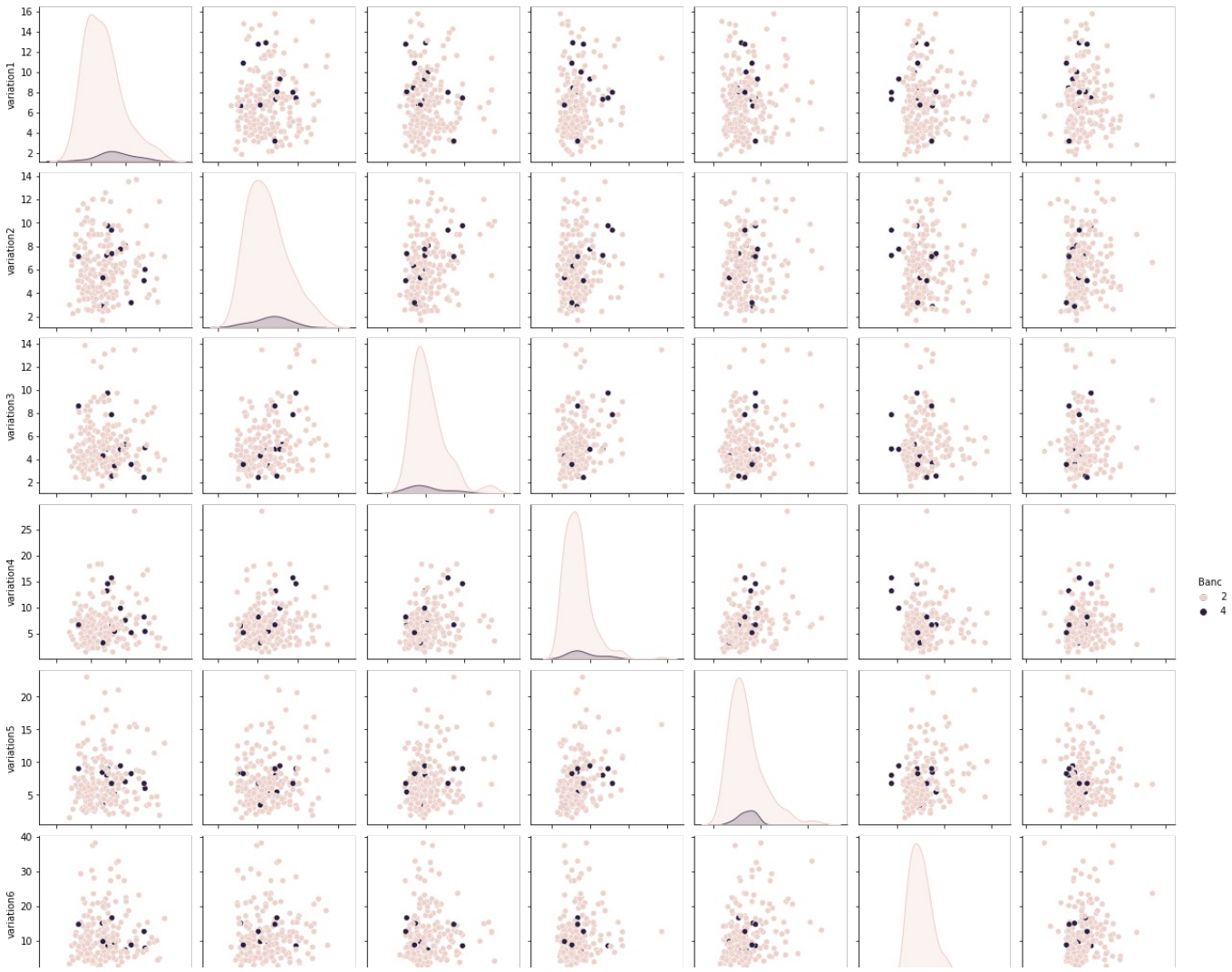
Echantillon	Banc	variation1	variation2	variation3	variation4	variation5	variation6	variation7
1301	4	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
A009	2	7.6250	6.6250	9.12500	13.3750	6.6250	23.750000	52.500000
A010	2	8.7500	7.3750	5.62500	3.5625	9.1875	32.625000	28.875000
A011	2	11.6250	13.6875	4.31250	3.0000	8.7500	8.750000	9.500000
A012	2	12.6000	7.1250	7.27500	16.3125	8.4375	6.750000	26.500000
...	...	...	...	...	...	...	...	...
T003	2	7.6250	4.7500	5.00000	3.8125	2.0625	5.250000	3.875000
T004	2	4.8750	7.5000	7.25000	5.3125	3.5625	7.875000	7.000000
T005	4	8.0000	9.3750	7.87500	15.7500	6.7500	-2.250000	10.500000
T007	4	10.8750	3.1875	3.56250	5.2500	8.2500	8.875000	3.125000
T008	4	3.1875	7.1250	8.62500	6.7500	9.0000	14.812500	4.687500

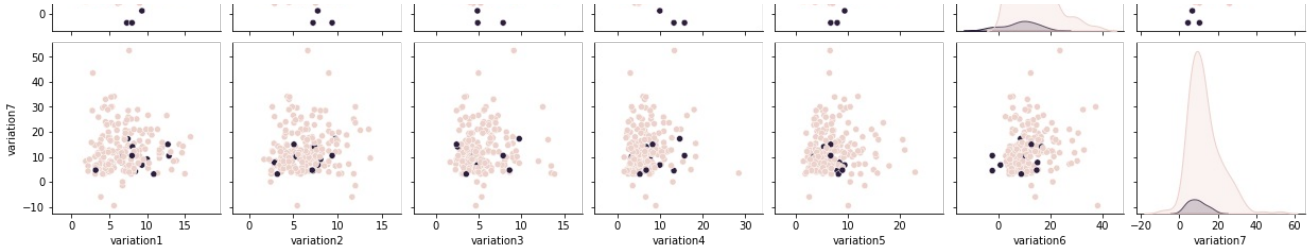
205 rows × 8 columns

A l'aide de la fonction "groupby" ,on regroupe les echantions puis on calcule leurs moyennes.

In [80]:
```
sns.pairplot(df,hue ='Banc')
```

Out[80]: <seaborn.axisgrid.PairGrid at 0x1ee54225670>

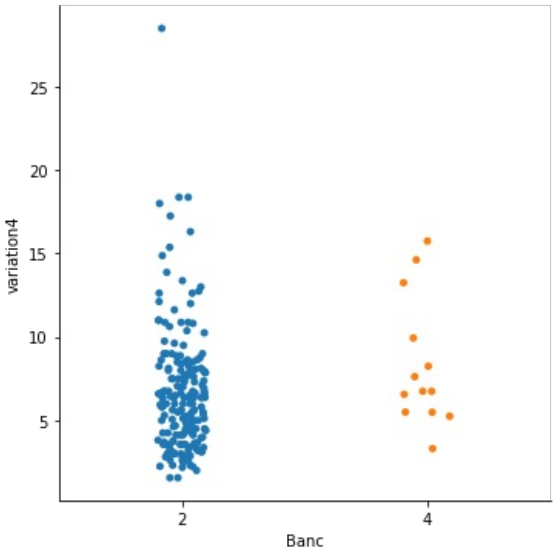




Ce graphique nous permet de bien visualiser les relations entre les variations tout en séparant les catégories 'banc =2' et 'banc =4' On peut même penser que le choix du banc n'a pas d'effets sur la germination car on visualise clairement que nos point s'interposent.

In [38]:
```
sns.catplot(x='Banc' , y='variation4' , data = df)
```

Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x1ee4d9243d0>
```



In [39]:
```
df[df["Banc"] == 4].shape
```

Out[39]:
```
(13, 8)
```

Analyse en Composantes Principales (Banc) :

In [40]:
```
#traitement de var. quali supplémentaire
vsQuali =  df.iloc[:,0]
print(vsQuali)
```

```
Echantillon
1301    4
A009    2
A010    2
A011    2
A012    2
       ..
T003    2
T004    2
T005    4
T007    4
T008    4
Name: Banc, Length: 205, dtype: int64
```

In [41]:
```
X = df[['variation1', 'variation2', 'variation3', 'variation4',
       'variation5', 'variation6', 'variation7']]
X.head()
```

Out[41]:

	variation1	variation2	variation3	variation4	variation5	variation6	variation7

Echantillon							
1301	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
A009	7.6250	6.6250	9.12500	13.3750	6.6250	23.750000	52.500000
A010	8.7500	7.3750	5.62500	3.5625	9.1875	32.625000	28.875000
A011	11.6250	13.6875	4.31250	3.0000	8.7500	8.750000	9.500000
A012	12.6000	7.1250	7.27500	16.3125	8.4375	6.750000	26.500000

In [42]:
```
#modalités de la variable qualitative
modalites = np.unique(vsQuali)
print(modalites)
```

```
[2 4]
```

In [43]:
```
(n,p)= X.shape
```

In [44]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [45]:
```
scaler  = StandardScaler()
Z = scaler.fit_transform(X)
```

In [46]:
```
mypca =  PCA()
mypca.fit(Z)
print(mypca.explained_variance_ratio_)
print(mypca.singular_values_)
print(mypca.components_)
```

```
[0.25913555 0.16163296 0.14415347 0.1423212  0.11847867 0.08975442
 0.08452373]
[19.28365919 15.22968457 14.38263636 14.2909383  13.03905275 11.34890299
 11.01324448]
[[ 0.15484009  0.36461102  0.47028892  0.47869646  0.50088653  0.30765244
   0.21731094]
 [-0.09552989 -0.26179567 -0.40952648 -0.20940923  0.18650195  0.6765818
   0.46714773]
 [-0.43008032  0.53982537  0.16696188 -0.27070146 -0.3190783  -0.12754619
   0.55171093]
 [ 0.85150808  0.22152865 -0.21344641 -0.04473119 -0.26851067 -0.08102169
   0.31565218]
 [-0.1064835  -0.5115894   0.00788372  0.54907689 -0.13055709 -0.36462061
   0.52478245]
 [ 0.1525191  -0.37497451  0.68919227 -0.2038969  -0.44302293  0.35113846
   0.00213801]
 [ 0.14920686 -0.23602088  0.25026509 -0.55556993  0.57193207 -0.41049737
   0.23479036]]
```

In [47]:
```
data_sortie= mypca.fit_transform(Z)
```

In [48]:
```
eigval = (n-1)/n*mypca.explained_variance_
print(eigval)
```

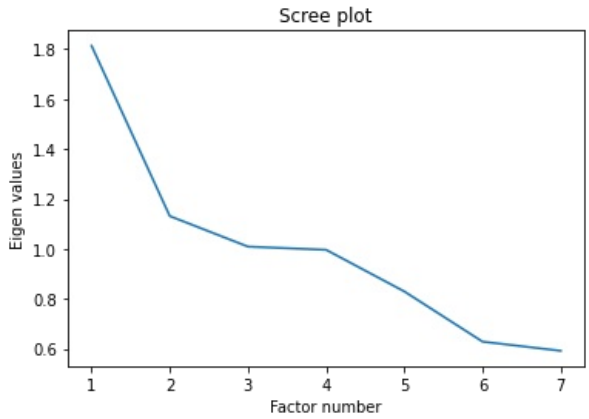
```
[1.81394884 1.13143069 1.00907429 0.99624838 0.82935072 0.62828097
 0.59166612]
```

In [49]:
```
print(mypca.explained_variance_ratio_)
```

```
[0.25913555 0.16163296 0.14415347 0.1423212  0.11847867 0.08975442
 0.08452373]
```

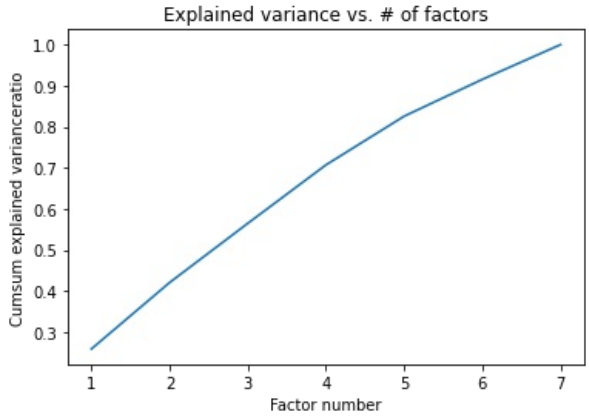
In [50]:
```
#scree plot
plt.plot(np.arange(1,p+1),eigval)
```

```
plt.title("Scree plot")
plt.ylabel("Eigen values")
plt.xlabel("Factor number")
plt.show()
```



In [51]:

```
#cumul de variance expliquée
plt.plot(np.arange(1,p+1),np.cumsum(mypca.explained_variance_ratio_))
plt.title("Explained variance vs. # of factors")
plt.ylabel("Cumsum explained varianceratio")
plt.xlabel("Factor number")
plt.show()
```



In [52]:

```
#seuils pour test des bâtons brisés
bs = 1/np.arange(p,0,-1)
bs = np.cumsum(bs)
bs = bs[::-1]
```

In [53]:

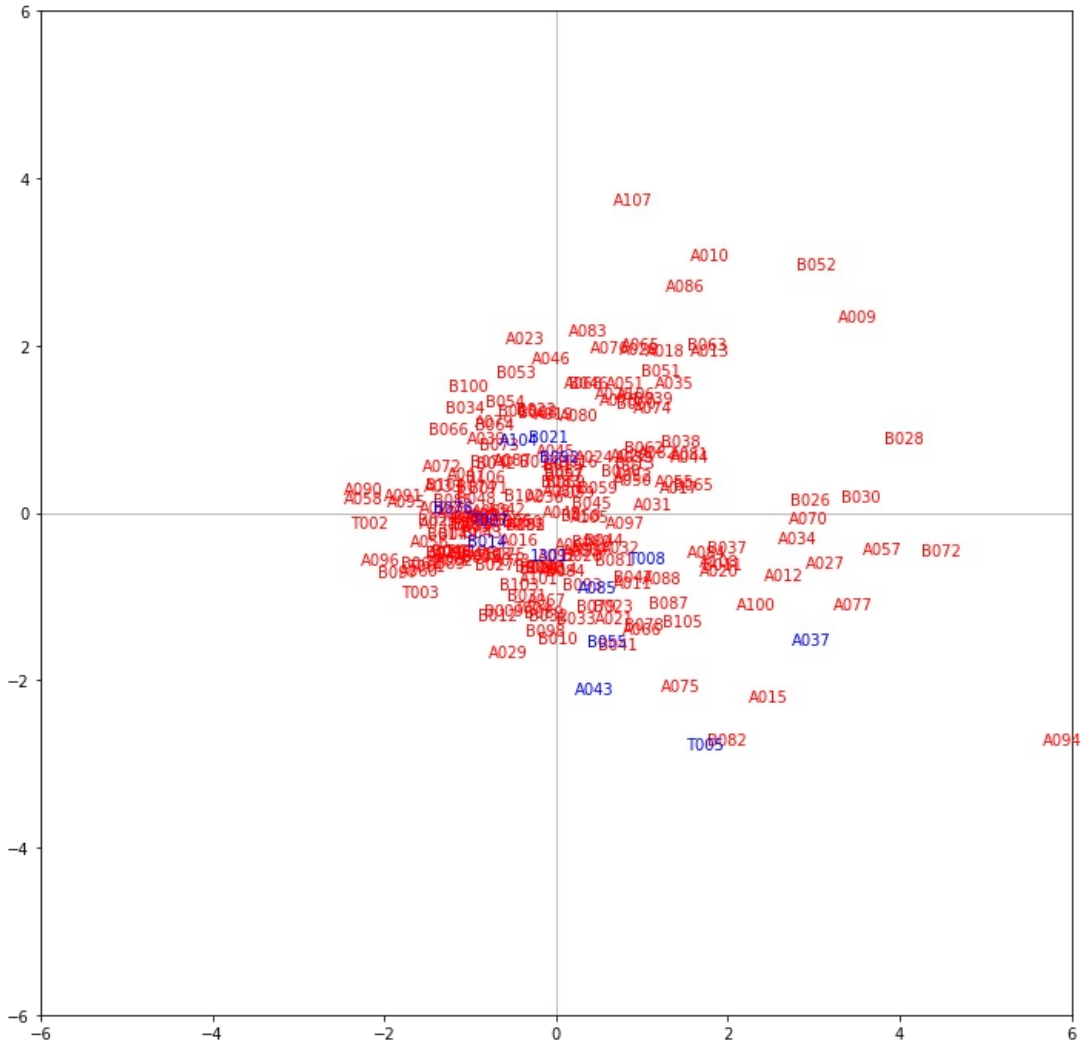
```
print(pd.DataFrame({'Val.Propre':eigval,'Seuils':bs}))
```

```
   Val.Propre    Seuils
0    1.813949  2.592857
1    1.131431  1.592857
2    1.009074  1.092857
3    0.996248  0.759524
4    0.829351  0.509524
5    0.628281  0.309524
6    0.591666  0.142857
```

In [54]:

```
#liste des couleurs
couleurs = ['r','b']
#faire un graphique en coloriant les points
fig, axes = plt.subplots(figsize=(12,12))
axes.set_xlim(-6,6)
axes.set_ylim(-6,6)
#pour chaque modalité de la var. illustrative
for c in range(len(modalites)):
    #numéro des individus concernés
    numero = np.where(vsQuali == modalites[c])
    #les passer en revue pour affichage
    for i in numero[0]:
```

```
        plt.annotate(X.index[i],(data_sortie[i,0],data_sortie[i,1]),color=couleurs[c])
#ajouter les axes
plt.plot([-6,6],[0,0],color='silver',linestyle='-',linewidth=1)
plt.plot([0,0],[-6,6],color='silver',linestyle='-',linewidth=1)
#affichage
plt.show()
```



Effet du choix de la zone sur la germination ?

Visualisation des données :

In [81]:
```
df2.head()
```

Out[81]:

	Echantillon	Banc	variation1	variation2	variation3	variation4	variation5	variation6	variation7
0	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
1	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
2	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
3	1301	4	12.75	7.95	6.000	5.925	8.475	6.15	8.25
4	1301	4	11.00	5.80	4.575	4.875	5.550	10.20	13.50

In [82]:
```
dfx = pd.read_excel("C:/Users/ua/Desktop/betterave.xlsx","Plan_semis")
#dfx.head()
dfxx = dfx["zone"]
#df2=df.copy()
df2 =pd.concat([df2, dfxx], axis=1)
df2.head()
```

Out[82]:

	Echantillon	Banc	variation1	variation2	variation3	variation4	variation5	variation6	variation7	zone
0	1301	4.0	12.75	7.95	6.000	5.925	8.475	6.15	8.25	14

1	1301	4.0	12.75	7.95	6.000	5.925	8.475	6.15	8.25	24
2	1301	4.0	12.75	7.95	6.000	5.925	8.475	6.15	8.25	34
3	1301	4.0	12.75	7.95	6.000	5.925	8.475	6.15	8.25	44
4	1301	4.0	11.00	5.80	4.575	4.875	5.550	10.20	13.50	14

In [83]:
```
df2 = df2.drop(["Banc"],axis =1)
df2.head()
```

Out[83]:

	Echantillon	variation1	variation2	variation3	variation4	variation5	variation6	variation7	zone
0	1301	12.75	7.95	6.000	5.925	8.475	6.15	8.25	14
1	1301	12.75	7.95	6.000	5.925	8.475	6.15	8.25	24
2	1301	12.75	7.95	6.000	5.925	8.475	6.15	8.25	34
3	1301	12.75	7.95	6.000	5.925	8.475	6.15	8.25	44
4	1301	11.00	5.80	4.575	4.875	5.550	10.20	13.50	14

In [85]:
```
df2["zone"].unique()
```

Out[85]:
```
array([14, 24, 34, 44, 32, 13, 22, 41, 43, 21, 11, 42, 31, 23, 12, 33],
      dtype=int64)
```

In [98]:
```
df3 = df2.groupby(["Echantillon","zone"]).mean()
df3.head(10)
```

Out[98]:

		variation1	variation2	variation3	variation4	variation5	variation6	variation7
Echantillon	**zone**							
1301	**14**	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	24	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	34	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	44	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
A009	**24**	7.6250	6.6250	9.12500	13.3750	6.6250	23.750000	52.500000
A010	**32**	8.7500	7.3750	5.62500	3.5625	9.1875	32.625000	28.875000
A011	**13**	11.6250	13.6875	4.31250	3.0000	8.7500	8.750000	9.500000
A012	**22**	12.6000	7.1250	7.27500	16.3125	8.4375	6.750000	26.500000
A013	**41**	3.5000	5.1250	6.75000	7.6875	12.4375	29.375000	11.625000
A014	**24**	11.8750	8.2500	3.50000	8.2500	4.5000	7.250000	11.500000

In [112…
```
df3.index[6][1]
df3.shape[0]
df3.iloc[1,7]
```

Out[112… nan

In [116…
```
df3["newzone"] = np.nan
for i in range(df3.shape[0]):
    df3.iloc[i,7] = str(df3.index[i][1])
df3.head(10)
```

Out[116…

		variation1	variation2	variation3	variation4	variation5	variation6	variation7	newzone
Echantillon	**zone**								
1301	**14**	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929	14
	24	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929	24
	34	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929	34
	44	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929	44

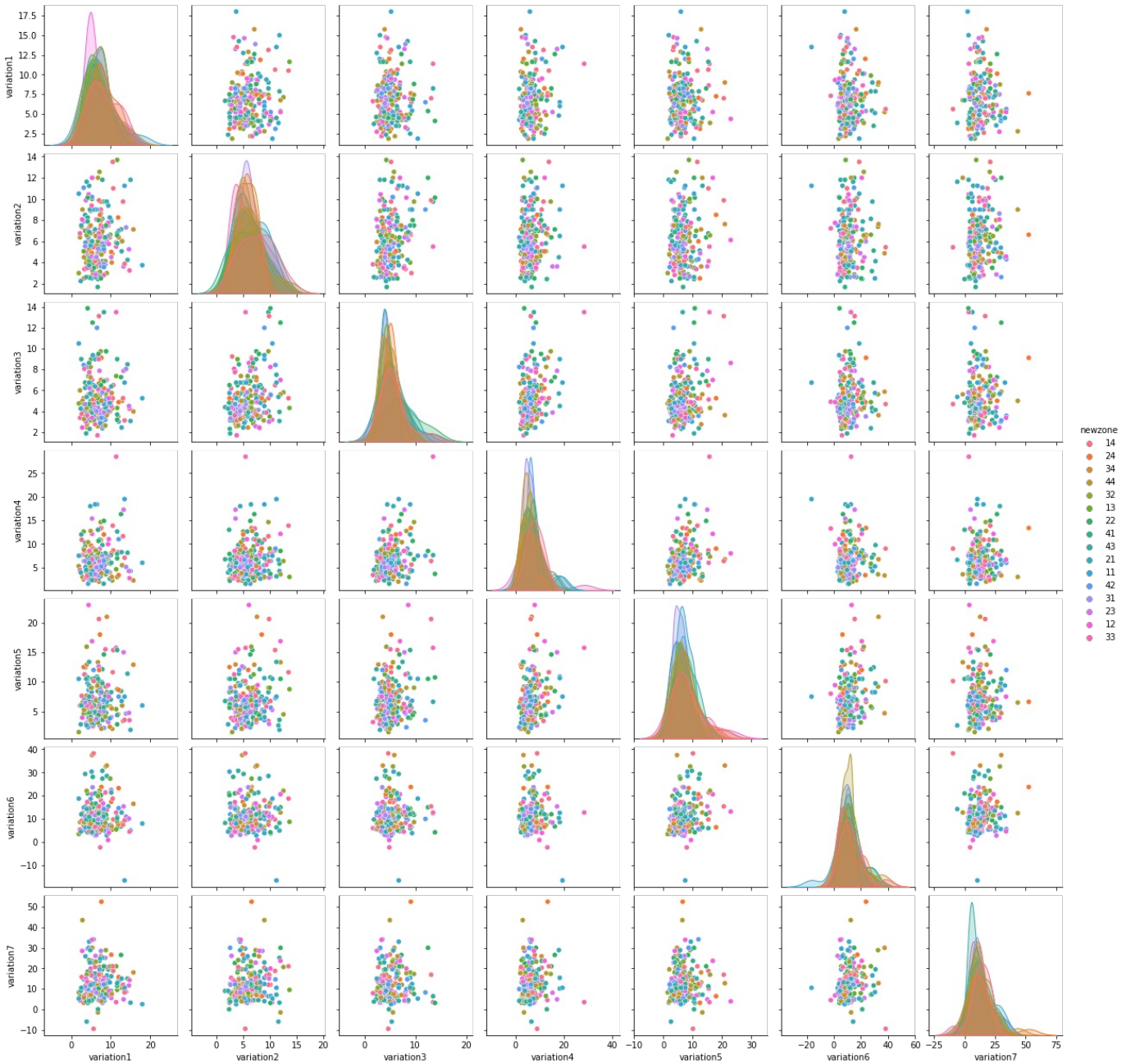
	A009	**24**	7.6250	6.6250	9.12500	13.3750	6.6250	23.750000	52.500000	24
	A010	**32**	8.7500	7.3750	5.62500	3.5625	9.1875	32.625000	28.875000	32
	A011	**13**	11.6250	13.6875	4.31250	3.0000	8.7500	8.750000	9.500000	13
	A012	**22**	12.6000	7.1250	7.27500	16.3125	8.4375	6.750000	26.500000	22
	A013	**41**	3.5000	5.1250	6.75000	7.6875	12.4375	29.375000	11.625000	41
	A014	**24**	11.8750	8.2500	3.50000	8.2500	4.5000	7.250000	11.500000	24

In [117…

```
sns.pairplot(df3,hue ="newzone")
```

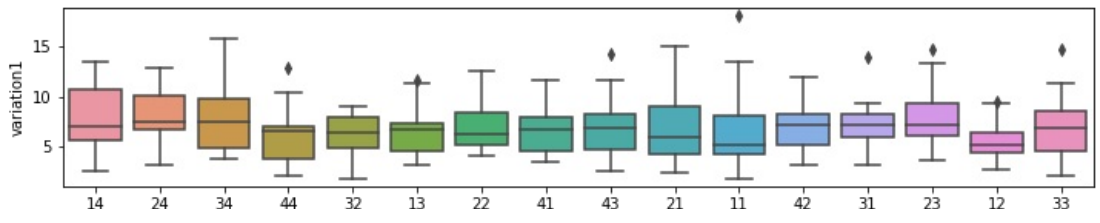
Out[117…

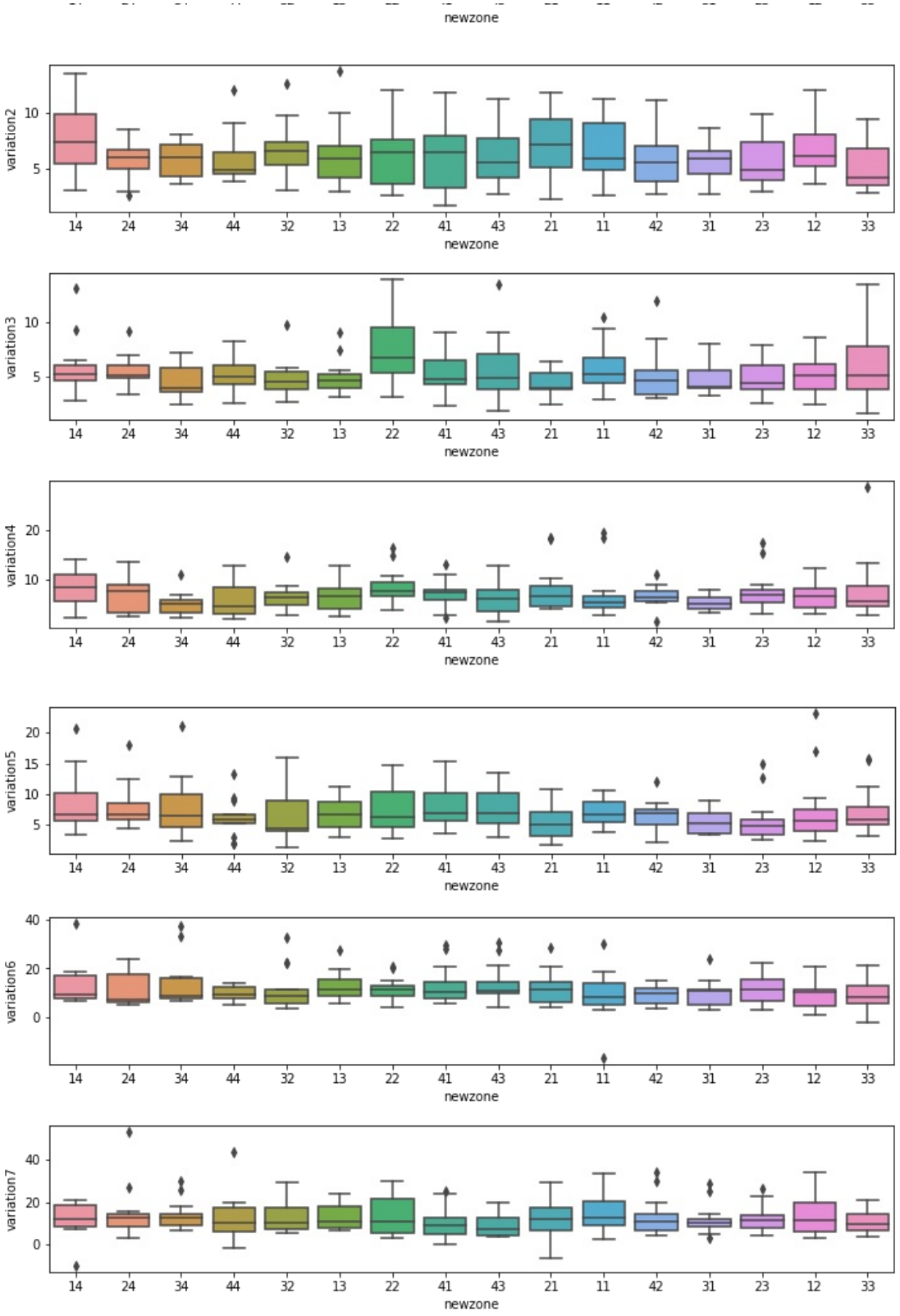
```
<seaborn.axisgrid.PairGrid at 0x1ee51a0b0a0>
```



In [131…

```
for i in range(1,8):
    plt.subplots(figsize=(12,20))
    plt.subplot(8,1,i)
    sns.boxplot(x='newzone' , y=f'variation{i}' , data = df3)
```





Analyse en Composantes principales (Zone) :

In [135…

```
#traitement de var. quali supplémentaire
vsQuali1 =  df3.iloc[:,7]
print(vsQuali)
```

```
Echantillon  zone
1301         14      14
             24      24
             34      34
             44      44
A009         24      24
                     ..
T005         43      43
```

```
T007          11      11
              43      43
T008          11      11
              43      43
Name: newzone, Length: 211, dtype: object
```

In [136…]
```
W = df3[['variation1', 'variation2', 'variation3', 'variation4',
       'variation5', 'variation6', 'variation7']]
W.head()
```

Out[136…]

		variation1	variation2	variation3	variation4	variation5	variation6	variation7
Echantillon	**zone**							
1301	**14**	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	24	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	34	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
	44	12.8875	6.0125	5.00625	5.4875	5.9875	7.984821	10.433929
A009	**24**	7.6250	6.6250	9.12500	13.3750	6.6250	23.750000	52.500000

In [138…]
```
#modalités de la variable qualitative
modalites1 = np.unique(vsQuali1)
print(modalites1)
```

```
['11' '12' '13' '14' '21' '22' '23' '24' '31' '32' '33' '34' '41' '42'
 '43' '44']
```

In [140…]
```
(n1 ,p1)= W.shape
```

In [141…]
```
scaler1  = StandardScaler()
Z1 = scaler1.fit_transform(W)
```

In [142…]
```
mypca1 =  PCA()
mypca1.fit(W)
print(mypca1.explained_variance_ratio_)
print(mypca1.singular_values_)
print(mypca1.components_)
```

```
[0.43839187 0.2817375  0.10917451 0.05813892 0.05196079 0.03902272
 0.02157368]
[123.71491232  99.17749552  61.73784034  45.05305786  42.59206517
  36.91046894  27.44435109]
[[-0.00274956  0.03979581  0.01609542  0.04695428  0.07895797  0.40763204
   0.90749475]
 [-0.01649748 -0.00934679  0.01657944  0.01327847  0.17636041  0.89146018
  -0.41639527]
 [ 0.15749836  0.16144953  0.23642497  0.74344182  0.56550112 -0.1394683
  -0.03581747]
 [-0.74346767 -0.02110333  0.04570459 -0.31258666  0.57716039 -0.11675824
   0.01626483]
 [ 0.64313597  0.05341869 -0.08605086 -0.5373376   0.53053063 -0.07516562
   0.01653797]
 [-0.04431626  0.87361557  0.41525856 -0.19104853 -0.1566      0.0198602
  -0.03122017]
 [ 0.08113303 -0.45360766  0.87271757 -0.14822812 -0.06140609  0.00258282
   0.01651097]]
```

In [143…]
```
data_sortie1= mypca1.fit_transform(Z1)
```

In [144…]
```
eigval1 = (n1-1)/n*mypca1.explained_variance_
print(eigval1)
```

```
[1.84761243 1.20830883 1.03297292 1.02417802 0.84209756 0.65447683
 0.59523145]
```

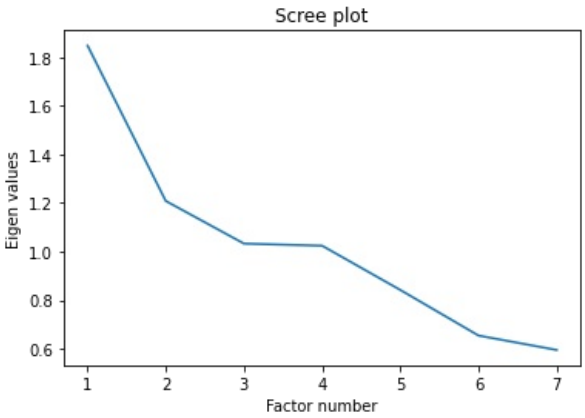
In [145…

```
print(mypca1.explained_variance_ratio_)
```

```
[0.2564391  0.16770705 0.14337133 0.14215064 0.11687881 0.09083802
 0.08261506]
```

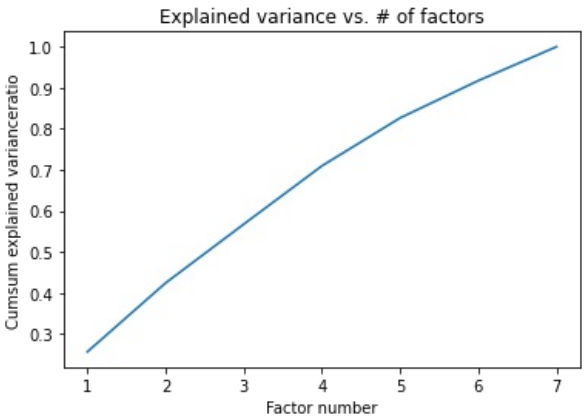
In [147…

```
#scree plot
plt.plot(np.arange(1,p1+1),eigval1)
plt.title("Scree plot")
plt.ylabel("Eigen values")
plt.xlabel("Factor number")
plt.show()
```



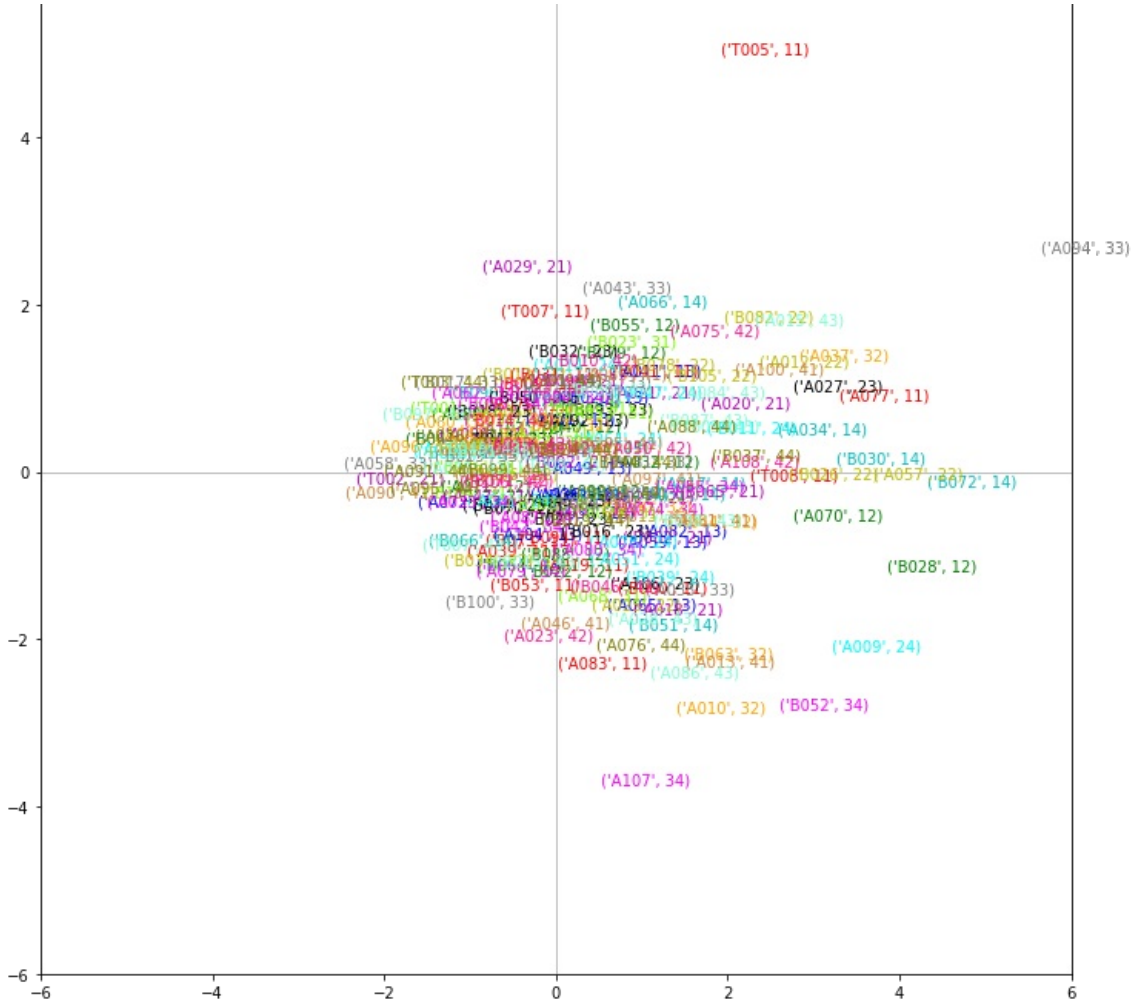
In [148…

```
#cumul de variance expliquée
plt.plot(np.arange(1,p1+1),np.cumsum(mypca1.explained_variance_ratio_))
plt.title("Explained variance vs. # of factors")
plt.ylabel("Cumsum explained varianceratio")
plt.xlabel("Factor number")
plt.show()
```



In [151…

```
#liste des couleurs
couleurs = ['r','g','b','c','m', 'y','k','aqua','lawngreen','orange','grey','magenta','peru','deeppink','aquamari
#faire un graphique en coloriant les points
fig, axes = plt.subplots(figsize=(12,12))
axes.set_xlim(-6,6)
axes.set_ylim(-6,6)
#pour chaque modalité de la var. illustrative
for c in range(len(modalites1)):
    #numéro des individus concernés
    numero = np.where(vsQuali1 == modalites1[c])
    #les passer en revue pour affichage
    for i in numero[0]:
        plt.annotate(W.index[i],(data_sortie1[i,0],data_sortie1[i,1]),color=couleurs[c])
#ajouter les axes
plt.plot([-6,6],[0,0],color='silver',linestyle='-',linewidth=1)
plt.plot([0,0],[-6,6],color='silver',linestyle='-',linewidth=1)
#affichage
plt.show()
```


('T005', 11)
('A094', 33)
('A029', 21)
('A043', 33)
('A066', 14)
('T007', 11)
('B055', 12)
('A075', 42)
('A027', 23)
('A020', 21)
('A077', 11)
('A088', 44)
('A034', 14)
('B030', 14)
('B072', 14)
('A070', 12)
('B028', 12)
('B053', 11)
('B100', 33)
('A046', 41)
('A023', 42)
('B051', 14)
('A076', 44)
('A009', 24)
('A083', 11)
('A010', 32)
('B052', 34)
('A107', 34)

In [ ]:

In [ ]:

In [ ]:

In [ ]: