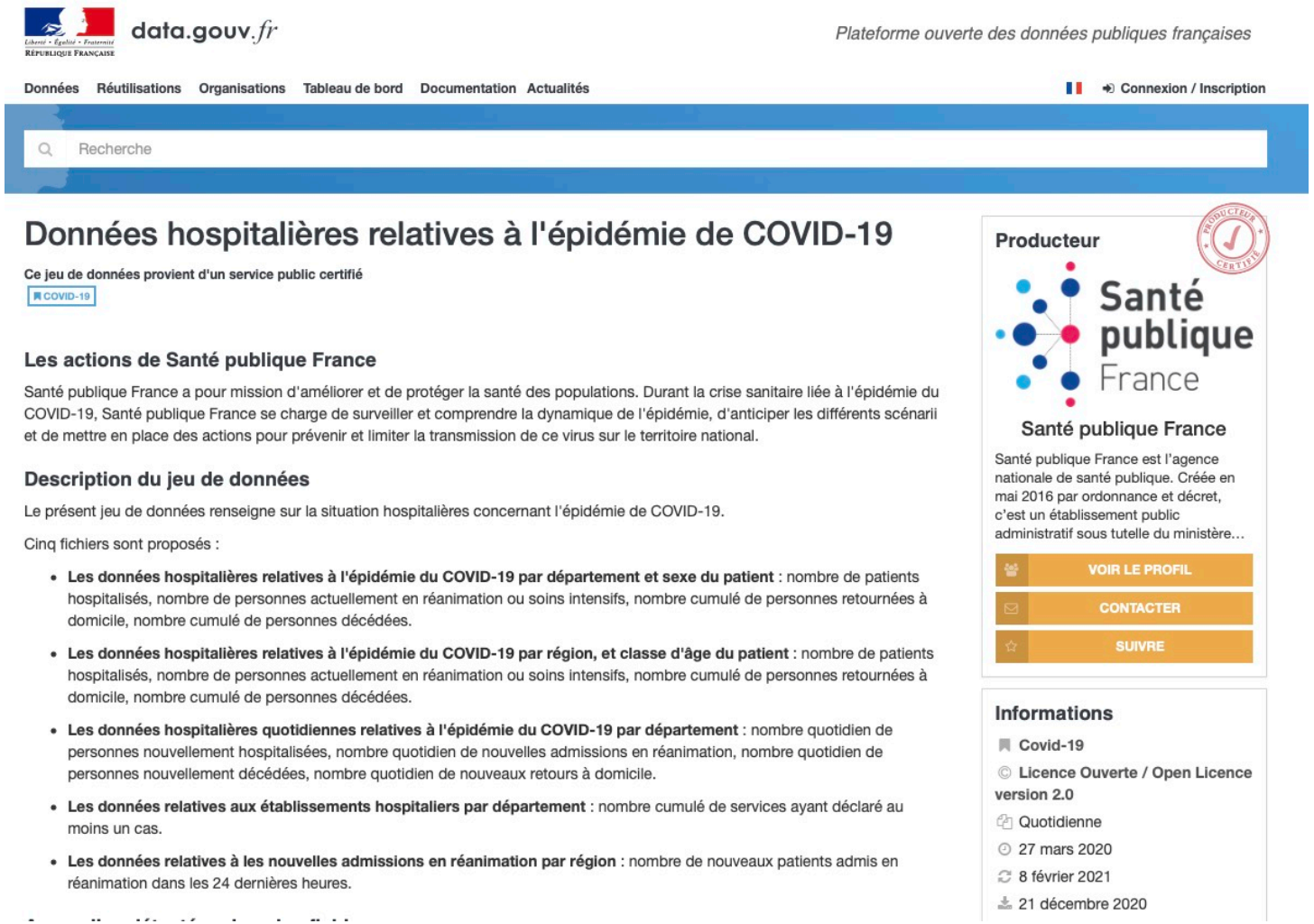
Aix*Marseille université
Socialement engagée

TP 1

Visualisation de séries temporelles avec Pandas et Matplotlib



data.gouv.fr

Plateforme ouverte des données publiques françaises

Données Réutilisations Organisations Tableau de bord Documentation Actualités

Connexion / Inscription

Recherche

Données hospitalières relatives à l'épidémie de COVID-19

Ce jeu de données provient d'un service public certifié

COVID-19

Les actions de Santé publique France

Santé publique France a pour mission d'améliorer et de protéger la santé des populations. Durant la crise sanitaire liée à l'épidémie du COVID-19, Santé publique France se charge de surveiller et comprendre la dynamique de l'épidémie, d'anticiper les différents scénarii et de mettre en place des actions pour prévenir et limiter la transmission de ce virus sur le territoire national.

Description du jeu de données

Le présent jeu de données renseigne sur la situation hospitalières concernant l'épidémie de COVID-19.

Cinq fichiers sont proposés :

- **Les données hospitalières relatives à l'épidémie du COVID-19 par département et sexe du patient** : nombre de patients hospitalisés, nombre de personnes actuellement en réanimation ou soins intensifs, nombre cumulé de personnes retournées à domicile, nombre cumulé de personnes décédées.
- **Les données hospitalières relatives à l'épidémie du COVID-19 par région, et classe d'âge du patient** : nombre de patients hospitalisés, nombre de personnes actuellement en réanimation ou soins intensifs, nombre cumulé de personnes retournées à domicile, nombre cumulé de personnes décédées.
- **Les données hospitalières quotidiennes relatives à l'épidémie du COVID-19 par département** : nombre quotidien de personnes nouvellement hospitalisées, nombre quotidien de nouvelles admissions en réanimation, nombre quotidien de personnes nouvellement décédées, nombre quotidien de nouveaux retours à domicile.
- **Les données relatives aux établissements hospitaliers par département** : nombre cumulé de services ayant déclaré au moins un cas.
- **Les données relatives à les nouvelles admissions en réanimation par région** : nombre de nouveaux patients admis en réanimation dans les 24 dernières heures.

Producteur

Santé publique France

Santé publique France est l'agence nationale de santé publique. Créée en mai 2016 par ordonnance et décret, c'est un établissement public administratif sous tutelle du ministère...

VOIR LE PROFIL

CONTACTER

SUIVRE

Informations

Covid-19

Licence Ouverte / Open Licence version 2.0

Quotidienne

27 mars 2020

8 février 2021

21 décembre 2020

https://www.data.gouv.fr/fr/datasets/donnees-hospitalieres-relatives-a-lepidemie-de-covid-19/

Ressources

Voir aussi : ressources communautaires

Fichier principal

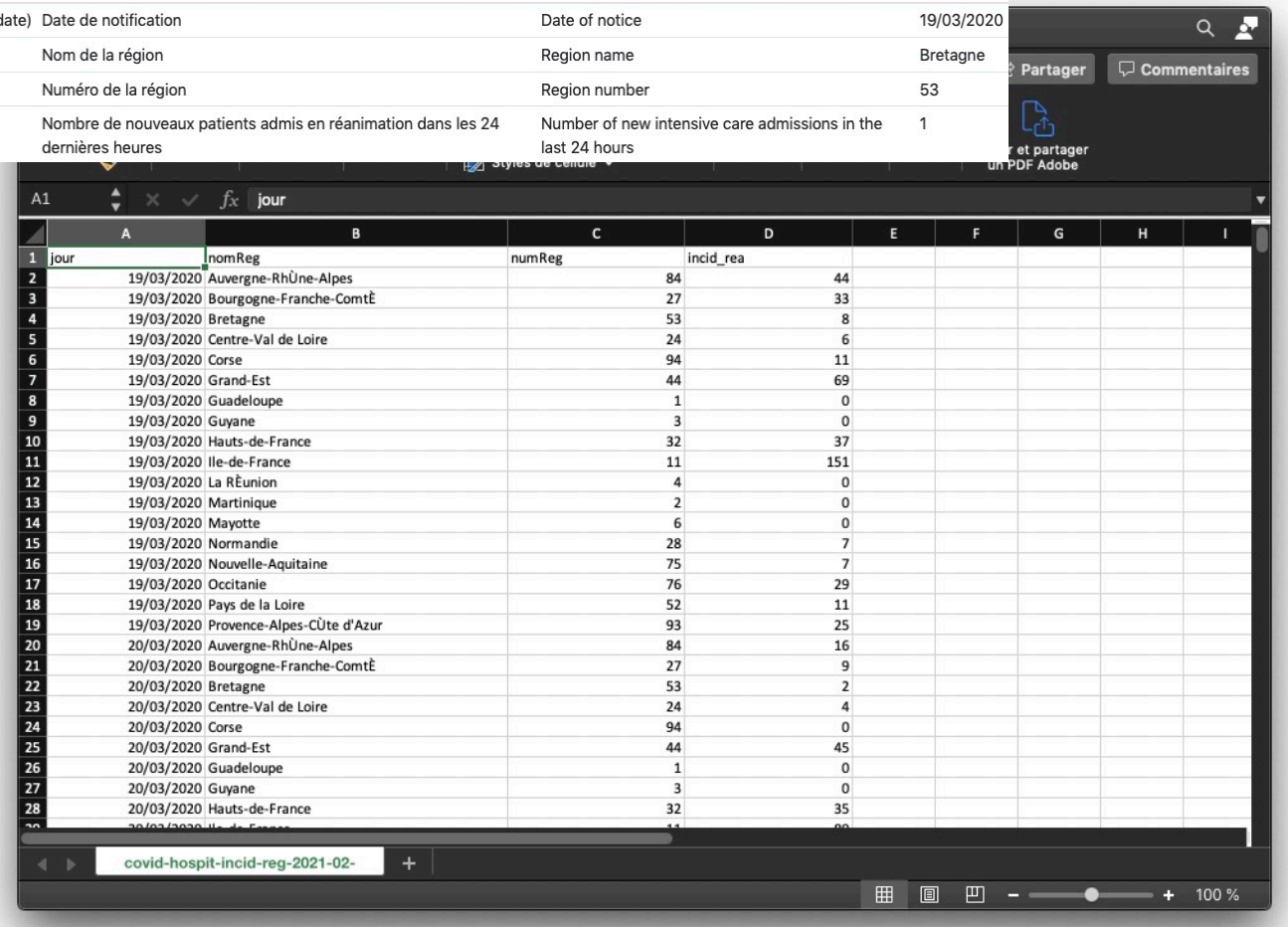
covid-hospit-incid-reg-2021-02-08-19h20.csv

csv (183.5Ko) 1517 Disponible

PRÉVISUALISER TÉLÉCHARGER

```
"jour";"nomReg";"numReg";"incid_rea"
2020-03-19;"Auvergne-RhÛne-Alpes";84;44
2020-03-19;"Bourgogne-Franche-ComtÈ";27;33
2020-03-19;"Bretagne";53;8
2020-03-19;"Centre-Val de Loire";24;6
2020-03-19;"Corse";94;11
2020-03-19;"Grand-Est";44;69
2020-03-19;"Guadeloupe";1;0
2020-03-19;"Guyane";3;0
2020-03-19;"Hauts-de-France";32;37
2020-03-19;"Ile-de-France";11;151
2020-03-19;"La RÈunion";4;0
2020-03-19;"Martinique";2;0
2020-03-19;"Mayotte";6;0
2020-03-19;"Normandie";28;7
2020-03-19;"Nouvelle-Aquitaine";75;7
2020-03-19;"Occitanie";76;29
2020-03-19;"Pays de la Loire";52;11
2020-03-19;"Provence-Alpes-CÛte d'Azur";93;25
2020-03-20;"Auvergne-RhÛne-Alpes";84;16
2020-03-20;"Bourgogne-Franche-ComtÈ";27;9
2020-03-20;"Bretagne";53;2
2020-03-20;"Centre-Val de Loire";24;4
2020-03-20;"Corse";94;0
2020-03-20;"Grand-Est";44;45
```

Colonne	Type	Description_FR	Description_EN	Exemple
jour	string($date)	Date de notification	Date of notice	19/03/2020
nomReg	string	Nom de la région	Region name	Bretagne
numReg	integer	Numéro de la région	Region number	53
incid_rea	integer	Nombre de nouveaux patients admis en réanimation dans les 24 dernières heures	Number of new intensive care admissions in the last 24 hours	1



	A	B	C	D
1	jour	nomReg	numReg	incid_rea
2	19/03/2020	Auvergne-RhÛne-Alpes	84	44
3	19/03/2020	Bourgogne-Franche-ComtÈ	27	33
4	19/03/2020	Bretagne	53	8
5	19/03/2020	Centre-Val de Loire	24	6
6	19/03/2020	Corse	94	11
7	19/03/2020	Grand-Est	44	69
8	19/03/2020	Guadeloupe	1	0
9	19/03/2020	Guyane	3	0
10	19/03/2020	Hauts-de-France	32	37
11	19/03/2020	Ile-de-France	11	151
12	19/03/2020	La RÈunion	4	0
13	19/03/2020	Martinique	2	0
14	19/03/2020	Mayotte	6	0
15	19/03/2020	Normandie	28	7
16	19/03/2020	Nouvelle-Aquitaine	75	7
17	19/03/2020	Occitanie	76	29
18	19/03/2020	Pays de la Loire	52	11
19	19/03/2020	Provence-Alpes-CÛte d'Azur	93	25
20	20/03/2020	Auvergne-RhÛne-Alpes	84	16
21	20/03/2020	Bourgogne-Franche-ComtÈ	27	9
22	20/03/2020	Bretagne	53	2
23	20/03/2020	Centre-Val de Loire	24	4
24	20/03/2020	Corse	94	0
25	20/03/2020	Grand-Est	44	45
26	20/03/2020	Guadeloupe	1	0
27	20/03/2020	Guyane	3	0
28	20/03/2020	Hauts-de-France	32	35

```
#%%
import numpy as np
import pandas as pd
# voir https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html
incidence = pd.read_csv('covid-hospit-incid-reg-2021-02-08-19h20.csv', sep=';', encoding="latin_1")
incidence.head()  incidence: {DataFrame: (5886, 5)}

#%%

incidence.dtypes  incidence: {DataFrame: (5886, 5)}

#%%
incidence['Date'] = incidence['jour'].astype("datetime64")  incidence: {DataFrame: (5886, 5)}  incidence: {Data
print(incidence.index.min())  incidence: {DataFrame: (5886, 5)}
print(incidence.index.max())  incidence: {DataFrame: (5886, 5)}
incidence.head()  incidence: {DataFrame: (5886, 5)}

#%%

incidence.drop(['jour'], axis=1, inplace=True)  incidence: {DataFrame: (5886, 5)}
incidence.head()  incidence: {DataFrame: (5886, 5)}
```

incidence['jour'] = pd.to_datetime(incidence['jour'])

51

	nomReg	numReg	incid_rea	Date
0	Auvergne-Rhône-Alpes	84	44	2020-03-19
1	Bourgogne-Franche-Comté	27	33	2020-03-19
2	Bretagne	53	8	2020-03-19
3	Centre-Val de Loire	24	6	2020-03-19
4	Corse	94	11	2020-03-19

```
#%%
print(incidence.groupby('nomReg')['incid_rea'].mean())  incidence: {DataFrame: (5886, 5)}
print(incidence.groupby('nomReg')['incid_rea'].sum())  incidence: {DataFrame: (5886, 5)}
print(incidence.groupby('nomReg')['incid_rea'].median())  incidence: {DataFrame: (5886, 5)}
```

nomReg	
Auvergne-Rhône-Alpes	22.492355
Bourgogne-Franche-Comté	7.669725
Bretagne	2.611621
Centre-Val de Loire	4.819572
Corse	0.541284
Grand-Est	16.446483
Guadeloupe	0.844037
Guyane	0.596330
Hauts-de-France	14.957187
Ile-de-France	45.348624
La Réunion	0.685015
Martinique	0.382263
Mayotte	0.391437
Normandie	4.911315
Nouvelle-Aquitaine	6.911315
Occitanie	10.899083
Pays de la Loire	4.516820
Provence-Alpes-Côte d'Azur	15.232416

nomReg	
Auvergne-Rhône-Alpes	7355
Bourgogne-Franche-Comté	2508
Bretagne	854
Centre-Val de Loire	1576
Corse	177
Grand-Est	5378
Guadeloupe	276
Guyane	195
Hauts-de-France	4891
Ile-de-France	14829
La Réunion	224
Martinique	125
Mayotte	128
Normandie	1606
Nouvelle-Aquitaine	2260
Occitanie	3564
Pays de la Loire	1477
Provence-Alpes-Côte d'Azur	4981

nomReg	
Auvergne-Rhône-Alpes	14
Bourgogne-Franche-Comté	4
Bretagne	1
Centre-Val de Loire	3
Corse	0
Grand-Est	9
Guadeloupe	0
Guyane	0
Hauts-de-France	10
Ile-de-France	31
La Réunion	0
Martinique	0
Mayotte	0
Normandie	3
Nouvelle-Aquitaine	4
Occitanie	7
Pays de la Loire	3
Provence-Alpes-Côte d'Azur	13

Extraction de 2 colonnes dans un nouveau DataFrame « regions » et regroupement des lignes selon le nom de la région :

```
regions = incidence[['incid_rea','nomReg']].groupby('nomReg')
```

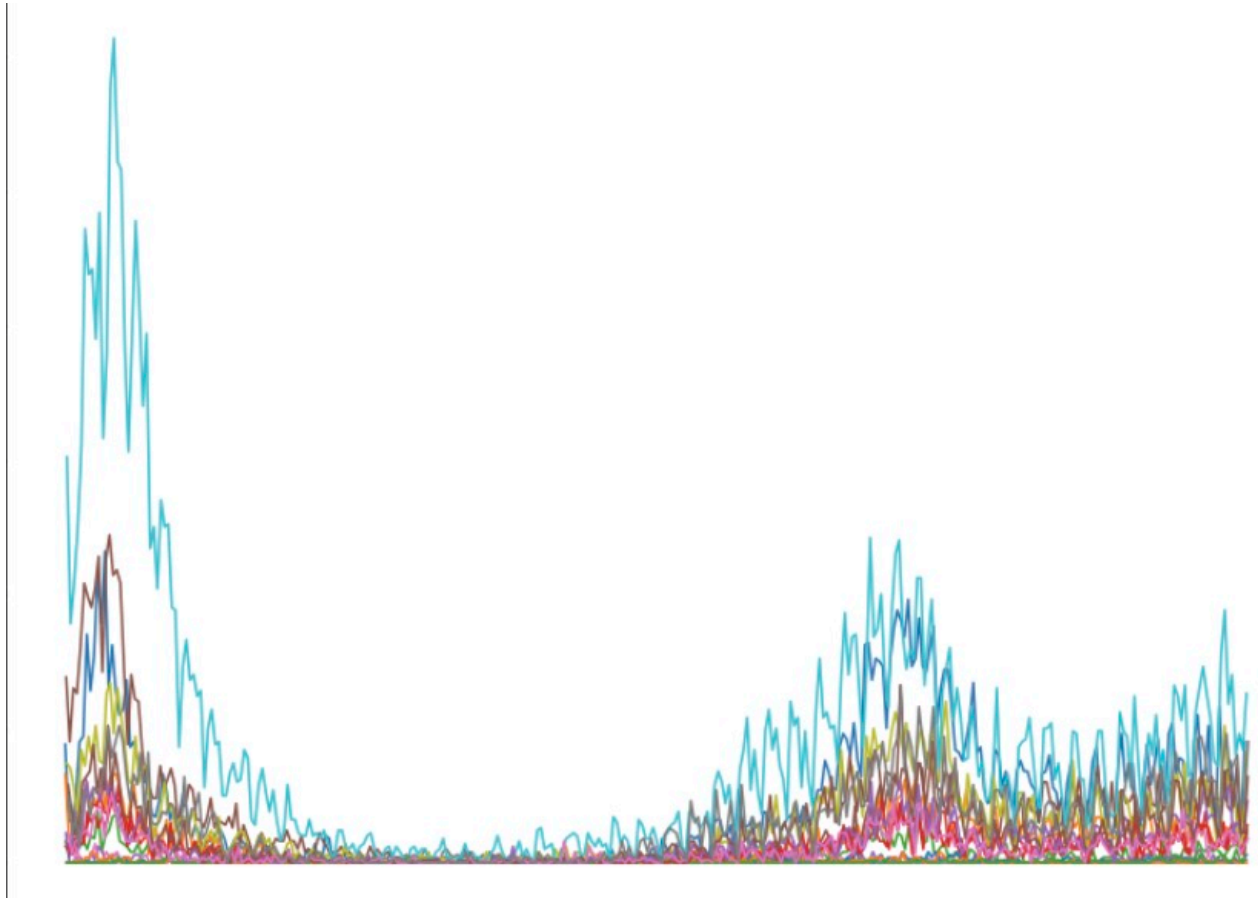
```
for nom, taux in regions:  regions: <pandas.core.gro
    print(nom)  nom: Provence-Alpes-Côte d'Azur
    print(taux)  taux: {DataFrame: (327, 2)}
```

```
Auvergne-Rhône-Alpes
      incid_rea                nomReg
0            44  Auvergne-Rhône-Alpes
18           16  Auvergne-Rhône-Alpes
36           15  Auvergne-Rhône-Alpes
54           25  Auvergne-Rhône-Alpes
72           45  Auvergne-Rhône-Alpes
...         ...                   ...
5796         38  Auvergne-Rhône-Alpes
5814         28  Auvergne-Rhône-Alpes
5832         20  Auvergne-Rhône-Alpes
5850          8  Auvergne-Rhône-Alpes
5868         45  Auvergne-Rhône-Alpes

[327 rows x 2 columns]
Bourgogne-Franche-Comté
      incid_rea                    nomReg
1            33  Bourgogne-Franche-Comté
19            9  Bourgogne-Franche-Comté
37           11  Bourgogne-Franche-Comté
55           14  Bourgogne-Franche-Comté
73           12  Bourgogne-Franche-Comté
...         ...                      ...
5797         10  Bourgogne-Franche-Comté
5815          9  Bourgogne-Franche-Comté
5833         11  Bourgogne-Franche-Comté
5851         19  Bourgogne-Franche-Comté
5869         22  Bourgogne-Franche-Comté

[327 rows x 2 columns]
```

```
for nom, taux in regions:  regions: <pandas.core.groupby.ge
    print(nom)  nom: Provence-Alpes-Côte d'Azur
    print(taux['incid_rea'])  taux: {DataFrame: (327, 2)}
```

```
Auvergne-Rhône-Alpes
0       44
18      16
36      15
54      25
72      45
        ..
5796    38
5814    28
5832    20
5850     8
5868    45
Name: incid_rea, Length: 327, dtype: int64
Bourgogne-Franche-Comté
1       33
19       9
37      11
55      14
73      12
        ..
5797    10
5815     9
5833    11
5851    19
5869    22
Name: incid_rea, Length: 327, dtype: int64
Bretagne
2       8
20      2
38      9
```

```
incidence.groupby('nomReg')['incid_rea'].plot(title='taux incidence', figsize=(14,10))
```

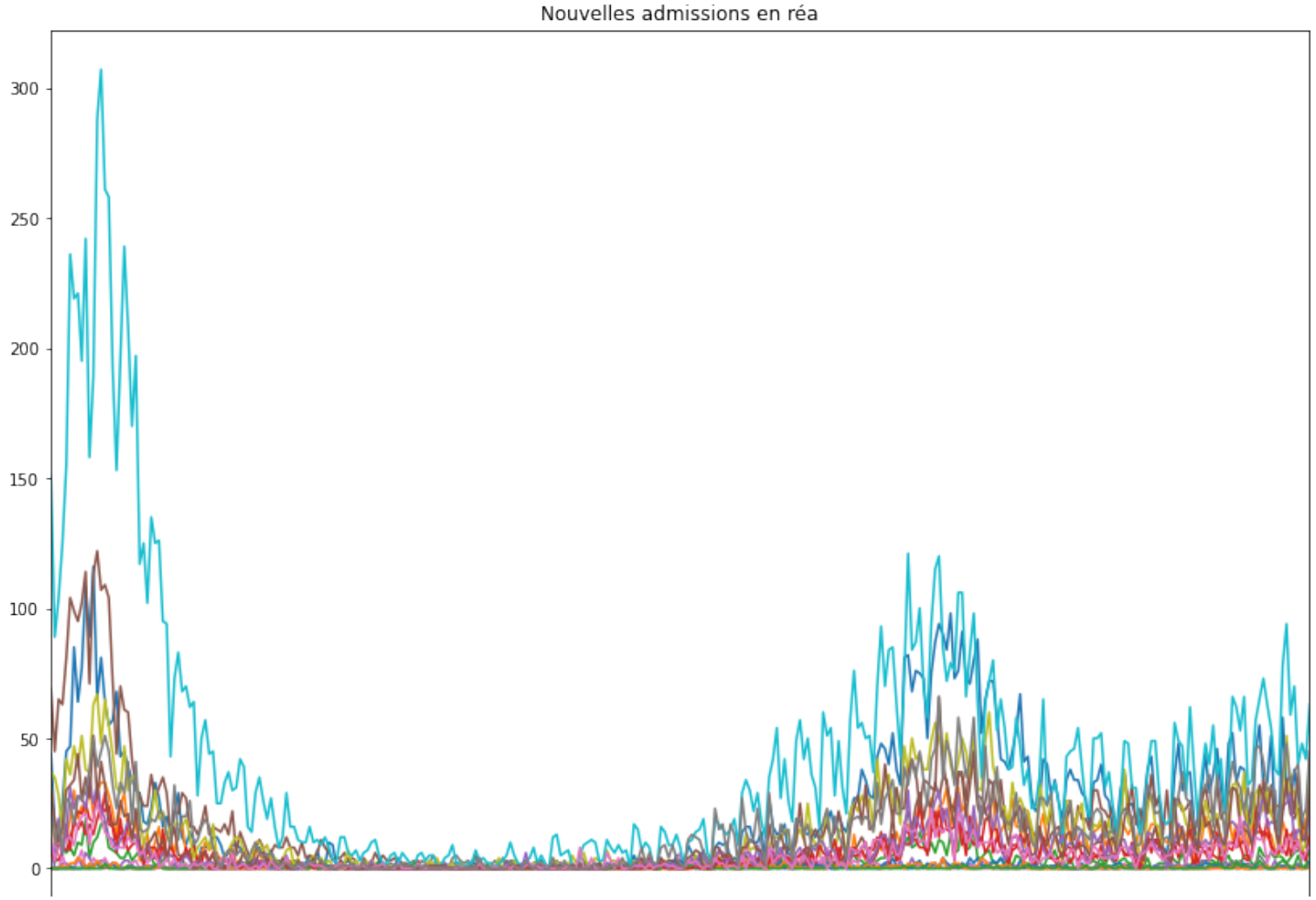


```
incidence['jour'] = incidence['jour'].astype("datetime64")  incidence: {DataFr
print(inc
incidence.set_index('date', inplace=True)
incidence.set_index('jour', inplace=True)  incidence: {DataFrame: (5886, 3)}

#%%

print("Première date : ", incidence.index.min())  incidence: {DataFrame: (588
print("Fin : ", incidence.index.max())  incidence: {DataFrame: (5886, 3)}
incidence.head()  incidence: {DataFrame: (5886, 3)}
```

```
Première date :  2020-03-19 00:00:00
Fin :  2021-02-08 00:00:00
```

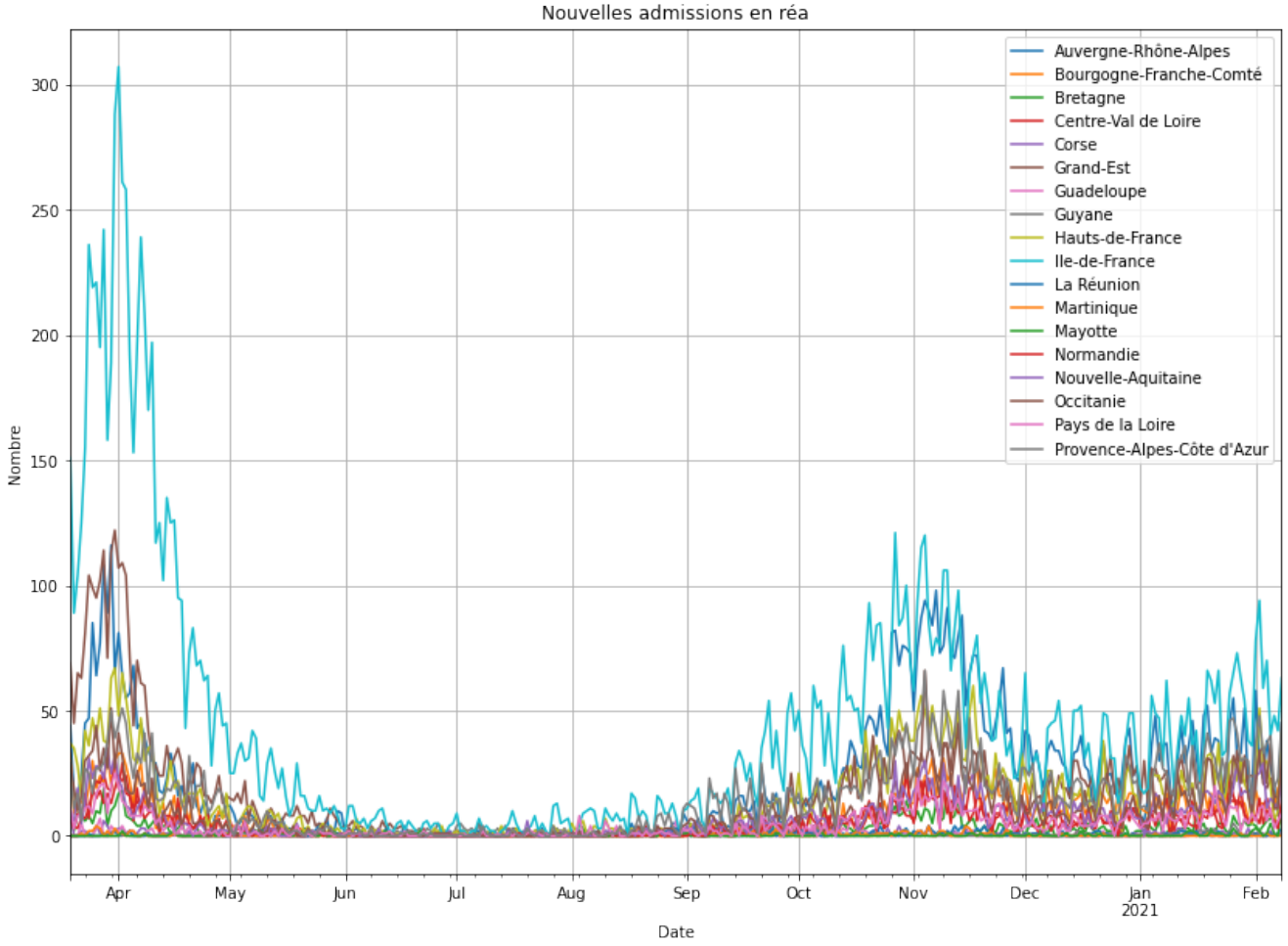
	nomReg	numReg	incid_rea
jour			
2020-03-19	Auvergne-Rhône-Alpes	84	44
2020-03-19	Bourgogne-Franche-Comté	27	33
2020-03-19	Bretagne	53	8
2020-03-19	Centre-Val de Loire	24	6
2020-03-19	Corse	94	11

```
incidence.groupby('nomReg')['incid_rea'].plot(title='Nouvelles admissions en réa', figsize=(14,10))
```



```
import matplotlib.pyplot as plt
```

```
df= incidence.groupby('nomReg')['incid_rea']  incidence: {Dat
df.plot(title='Nouvelles admissions en réa', figsize=(14,10))
plt.xlabel("Date")
plt.ylabel("Nombre")
plt.legend()
plt.grid(True)
```



Aix*Marseille université
Socialement engagée

TP2

Application d'une classification
(méthode des k-moyennes et classification hiérarchique)

Les données doivent être mises sous forme « une ligne = un individu »

```
Première date :  2020-03-19 00:00:00
Fin :  2021-02-08 00:00:00
```

	nomReg	numReg	incid_rea
jour			
2020-03-19	Auvergne-Rhône-Alpes	84	44
2020-03-19	Bourgogne-Franche-Comté	27	33
2020-03-19	Bretagne	53	8
2020-03-19	Centre-Val de Loire	24	6
2020-03-19	Corse	94	11

```
newDF = incidence.pivot(index='nomReg', columns='jour', values='incid_rea')
```

jour	2020-03-19	2020-03-20	2020-03-21	2020-03-22	2020-03-23	2020-03-24	2020-03-25	2020-03-26	2020-03-27	2020-03-28	...
nomReg											
Auvergne-Rhône-Alpes	44	16	15	25	45	47	85	64	77	108	...
Bourgogne-Franche-Comté	33	9	11	14	12	19	30	19	24	21	...
Bretagne	8	2	9	8	6	8	5	10	9	19	...
Centre-Val de Loire	6	4	3	7	17	7	19	15	18	13	...
Corse	11	0	0	2	2	0	0	4	1	6	...

Calculer une matrice de similarités 2 à 2

Deuxième étape : remplir la matrice avec les produits scalaires des vecteurs 2 à 2

```
creation_matrice_sim(temperatures)
```

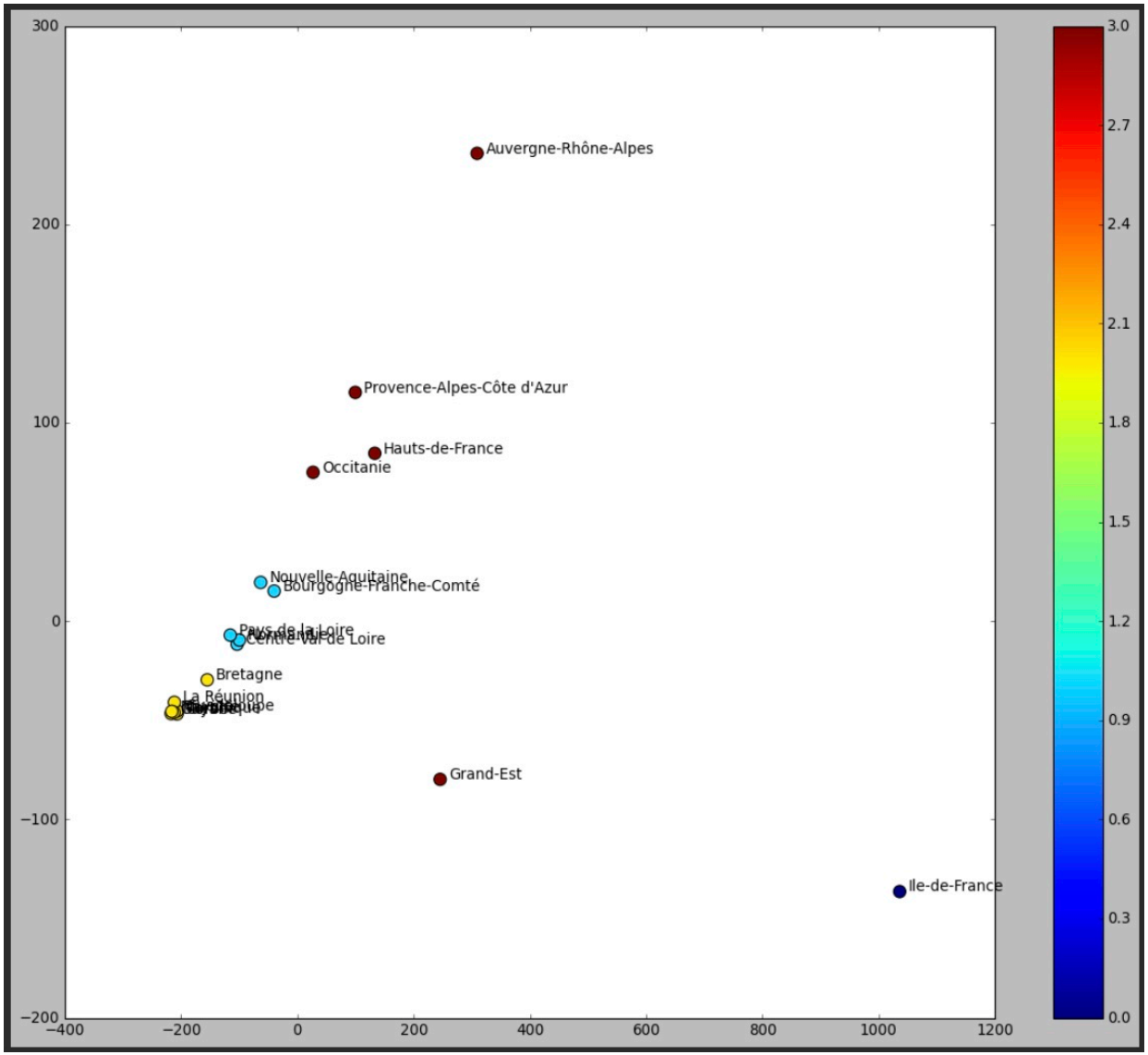
```
from sklearn.metrics import pairwise_distances
SIM = pairwise_distances(temperatures, metric=np.dot)

print("avec un cosinus : ")
COS = pairwise_distances(temperatures, metric='cosine')
```

```
#les autres fonctions prévues : https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.distance_metrics.html
```

```
from sklearn.cluster import KMeans
```

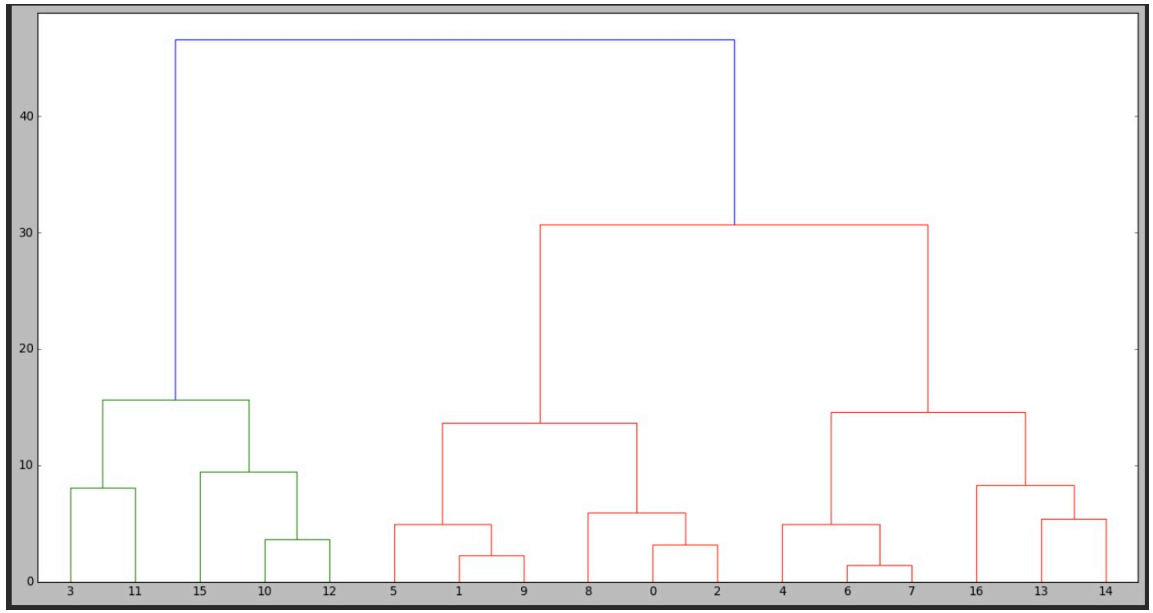
```
kmean_model = KMeans(n_clusters=4, n_init=2, max_iter=100, init="random")
%time km = kmean_model.fit_predict(newDF)
```

```
pca = PCA(n_components=3).fit(newDF)
data = pca.transform(newDF)
centroids = pca.transform(kmean_model.cluster_centers_)
```


Auvergne-Rhône-Alpes
Provence-Alpes-Côte d'Azur
Hauts-de-France
Occitanie
Nouvelle-Aquitaine
Bourgogne-Franche-Comté
Pays de la Loire
Centre-Val de Loire
Bretagne
La Réunion
Grand-Est
Ile-de-France
300
200
100
0
−100
−200
−400
−200
0
200
400
600
800
1000
1200
3.0
2.7
2.4
2.1
1.8
1.5
1.2
0.9
0.6
0.3
0.0

Construction d'une classification hiérarchique

```
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster import hierarchy

model = AgglomerativeClustering(distance_threshold=None, n_clusters=15)
model = model.fit(newDF)

Z = hierarchy.linkage(model.children_, 'ward')
plt.figure(figsize=(20,10))
dn = hierarchy.dendrogram(Z)
```



Aix*Marseille université
Socialement engagée

TP3

Analyse en composantes principales

Aix*Marseille université
Socialement engagée

dataset breast_cancer

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
cancer.keys()
```

```
261 dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])

262 cancer

262 {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n----------
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
```

```
df = pd.DataFrame(cancer['data'], columns=cancer['feature_names'])
df.head()
```

263

```
df = pd.DataFrame(cancer['data'], columns=cancer['feature_names'])
df.head()
```

263

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883

5 rows × 30 columns

```
264 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

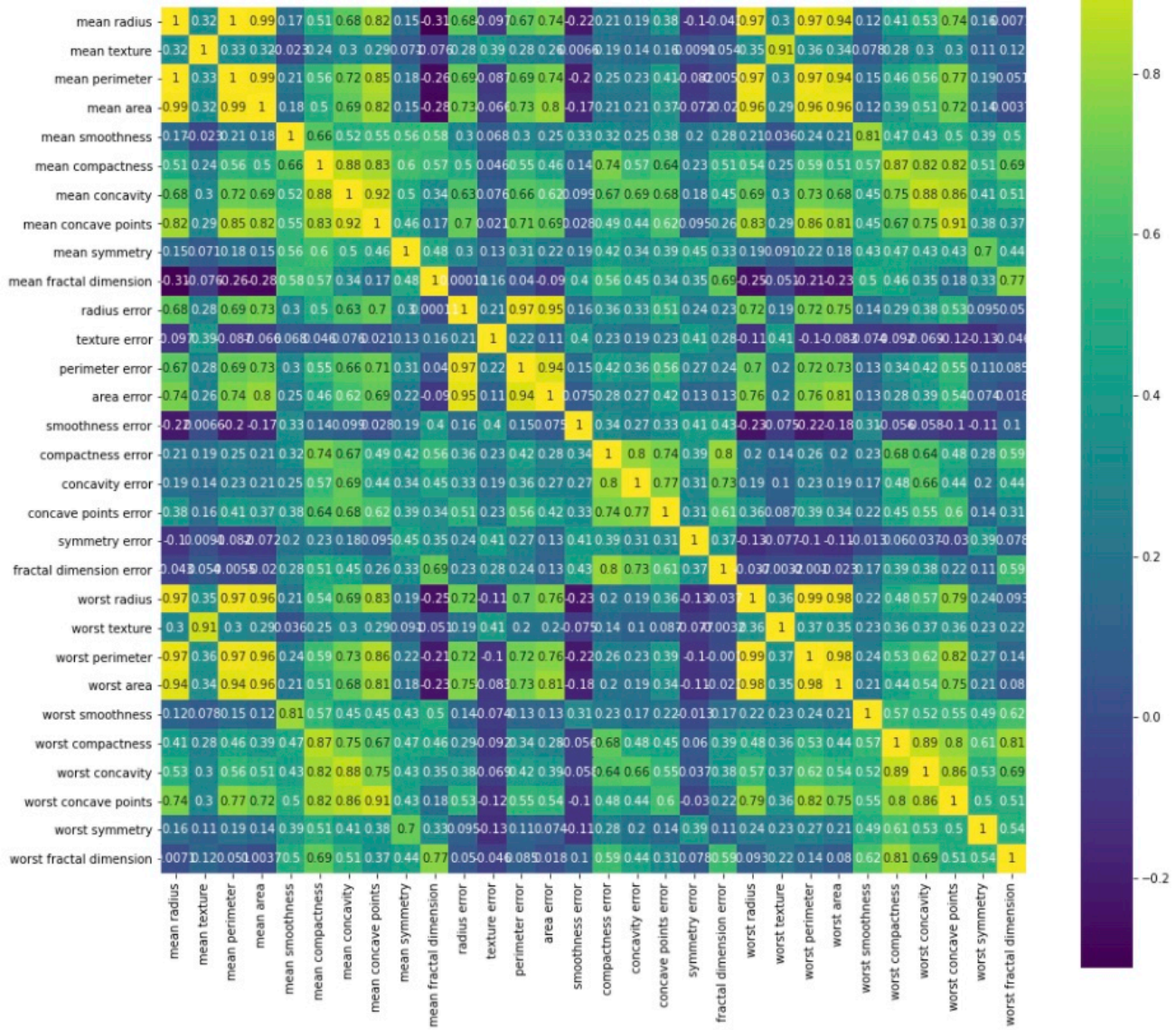
df.corr()

```
265 df.corr()
```

265

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
mean radius	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0
mean texture	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0
mean perimeter	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0
mean area	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0
mean smoothness	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0
mean compactness	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0
mean concavity	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0
mean concave points	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0
mean symmetry	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.
mean fractal dimension	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0
radius error	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0
texture error	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0
perimeter error	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0

```
plt.figure(figsize=(16,16))
sns.heatmap(df.corr(), vmax=1, square=True,annot=True,cmap='viridis')
```



```
280 from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    X=scaler.fit_transform(df)
    X
```

```
280 array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
             2.75062224,  1.93701461],
           [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
            -0.24388967,  0.28118999],
           [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
             1.152255  ,  0.20139121],
           ...,
           [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
            -1.10454895, -0.31840916],
           [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
             1.91908301,  2.21963528],
           [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
            -0.04813821, -0.75120669]])
```

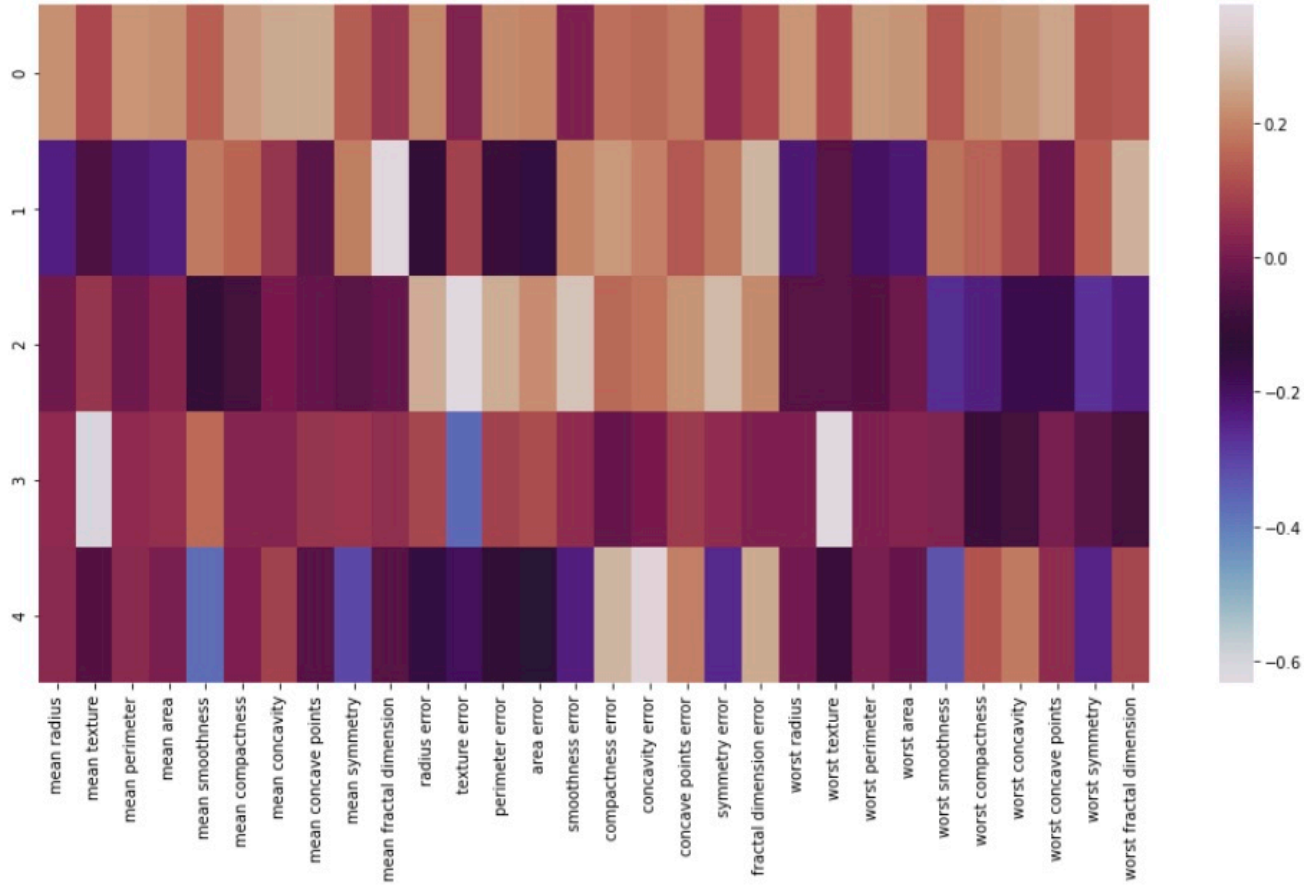
```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit_transform(X)
```

```
print(pca.components_)
```

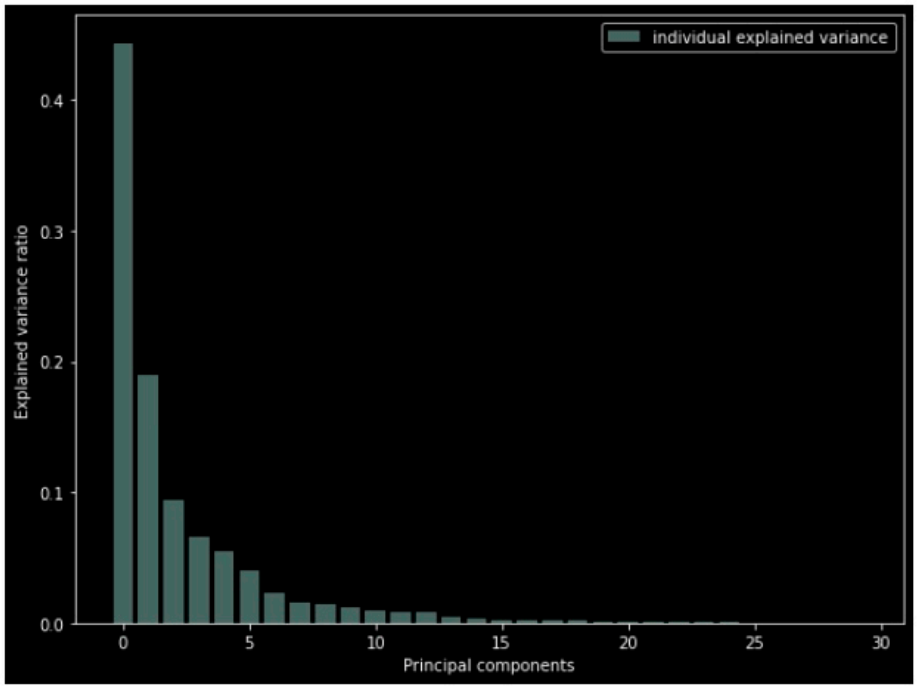
```
[[ 2.18902444e-01  1.03724578e-01  2.27537293e-01  2.20994985e-01
   1.42589694e-01  2.39285354e-01  2.58400481e-01  2.60853758e-01
   1.38166959e-01  6.43633464e-02  2.05978776e-01  1.74280281e-02
   2.11325916e-01  2.02869635e-01  1.45314521e-02  1.70393451e-01
   1.53589790e-01  1.83417397e-01  4.24984216e-02  1.02568322e-01
   2.27996634e-01  1.04469325e-01  2.36639681e-01  2.24870533e-01
   1.27952561e-01  2.10095880e-01  2.28767533e-01  2.50885971e-01
   1.22904556e-01  1.31783943e-01]
 [-2.33857132e-01 -5.97060883e-02 -2.15181361e-01 -2.31076711e-01
   1.86113023e-01  1.51891610e-01  6.01653628e-02 -3.47675005e-02
   1.90348770e-01  3.66575471e-01 -1.05552152e-01  8.99796818e-02
  -8.94572342e-02 -1.52292628e-01  2.04430453e-01  2.32715896e-01
   1.97207283e-01  1.30321560e-01  1.83848000e-01  2.80092027e-01
  -2.19866379e-01 -4.54672983e-02 -1.99878428e-01 -2.19351858e-01
   1.72304352e-01  1.43593173e-01  9.79641143e-02 -8.25723507e-03
   1.41883349e-01  2.75339469e-01]
 [-8.53124284e-03  6.45499033e-02 -9.31421972e-03  2.86995259e-02
  -1.04291904e-01 -7.40915709e-02  2.73383798e-03 -2.55635406e-02
  -4.02399363e-02 -2.25740897e-02  2.68481387e-01  3.74633665e-01
   2.66645367e-01  2.16006528e-01  3.08838979e-01  1.54779718e-01
   1.76463743e-01  2.24657567e-01  2.88584292e-01  2.11503764e-01
  -4.75069900e-02 -4.22978228e-02 -4.85465083e-02 -1.19023182e-02
  -2.59797613e-01 -2.36075625e-01 -1.73057335e-01 -1.70344076e-01
  -2.71312642e-01 -2.32791313e-01]
 [ 4.14089623e-02 -6.03050001e-01  4.19830991e-02  5.34337955e-02
   1.59382765e-01  3.17945811e-02  1.91227535e-02  6.53359443e-02
   6.71249840e-02  4.85867649e-02  9.79412418e-02 -3.59855528e-01
   8.89924146e-02  1.08205039e-01  4.46641797e-02 -2.74693632e-02
   1.31687997e-03  7.40673350e-02  4.40733510e-02  1.53047496e-02
   1.54172396e-02 -6.32807885e-01  1.38027944e-02  2.58947492e-02
   1.76522161e-02 -9.13284153e-02 -7.39511797e-02  6.00699571e-03
  -3.62506947e-02 -7.70534703e-02]
 [ 3.77863538e-02 -4.94688505e-02  3.73746632e-02  1.03312514e-02
  -3.65088528e-01  1.17039713e-02  8.63754118e-02 -4.38610252e-02
  -3.05941428e-01 -4.44243602e-02 -1.54456496e-01 -1.91650506e-01
```

```
map= pd.DataFrame(pca.components_,columns=cancer['feature_names'])
plt.figure(figsize=(12,6))
sns.heatmap(map,cmap='twilight')
```

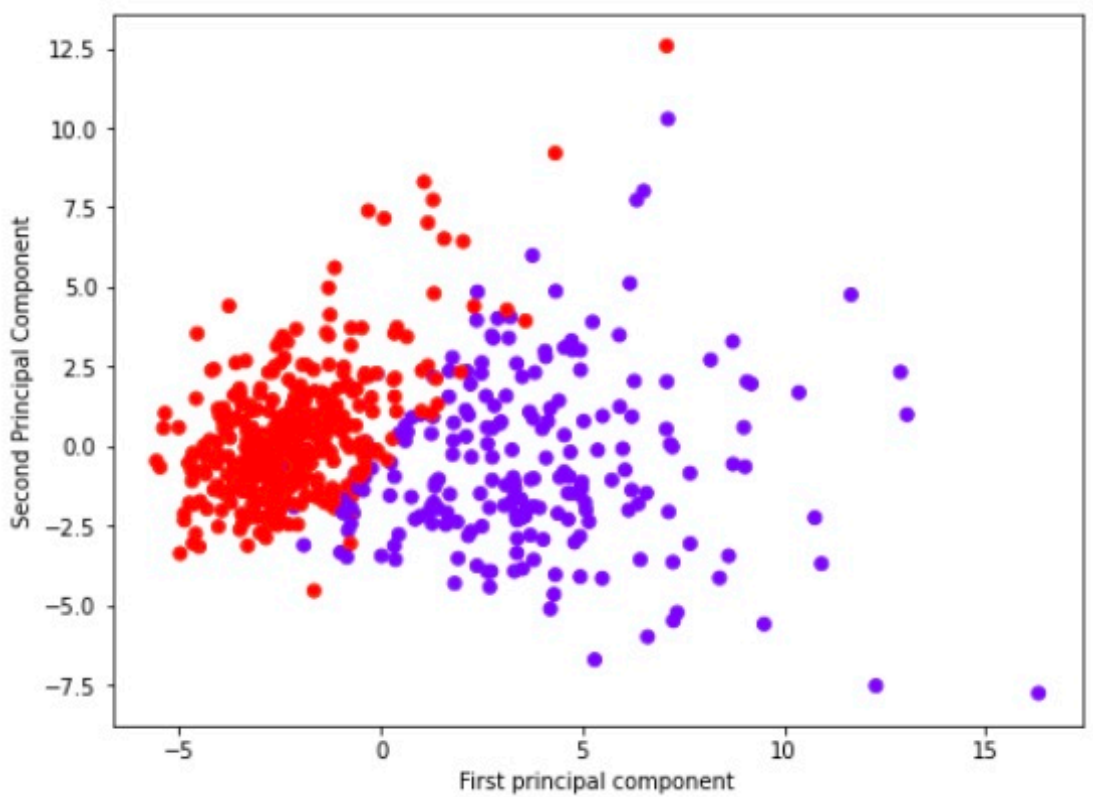
5 premières composantes



```
explained_variance=pca.explained_variance_ratio_
explained_variance
#%%
with plt.style.context('dark_background'):
    plt.figure(figsize=(6, 4))

    plt.bar(range(X[0].size), explained_variance, alpha=0.5, align='center',
            label='individual explained variance')
    plt.ylabel('Explained variance ratio')
    plt.xlabel('Principal components')
    plt.legend(loc='best')
```



```
pca=PCA(n_components=5)
X_new=pca.fit_transform(X)
plt.figure(figsize=(8,6))
plt.scatter(X_new[:,0],X_new[:,1],c=cancer['target'],cmap='rainbow')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```



et sur les données Titanic ?

```
df = pd.read_csv("titanic.csv")
df.info()
```

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
1	0	3	0	22.0	1	0	7.2500
2	1	1	1	38.0	1	0	71.2833
3	1	3	1	26.0	0	0	7.9250
4	1	1	1	35.0	1	0	53.1000
5	0	3	0	35.0	0	0	8.0500
7	0	1	0	54.0	0	0	51.8625
8	0	3	0	2.0	3	1	21.0750
9	1	3	1	27.0	0	2	11.1333
10	1	2	1	14.0	1	0	30.0708
11	1	3	1	4.0	1	1	16.7000
12	1	1	1	58.0	0	0	26.5500
13	0	3	0	20.0	0	0	8.0500
14	0	3	0	39.0	1	5	31.2750
15	0	3	1	14.0	0	0	7.8542
16	1	2	1	55.0	0	0	16.0000
17	0	3	0	2.0	4	1	29.1250
19	0	3	1	31.0	1	0	18.0000
21	0	2	0	35.0	0	0	26.0000
22	1	2	0	34.0	0	0	13.0000

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
st = sc.fit_transform(titanicSansNA.iloc[:,[3,6]])
titanicStd = titanicSansNA.copy()
titanicStd.iloc[:,[3,6]] = pd.DataFrame(st,index=titanicSansNA.index,columns=titanicSansNA.columns[[3,6]])
```

```
titanicStd
```

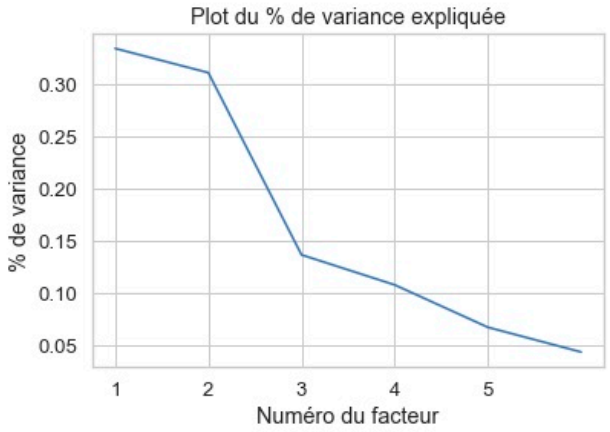
PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
1	0	3	0	-0.530377	1	0	-0.518978
2	1	1	1	0.571831	1	0	0.691897
3	1	3	1	-0.254825	0	0	-0.506214
4	1	1	1	0.365167	1	0	0.348049
5	0	3	0	0.365167	0	0	-0.503850
7	0	1	0	1.674039	0	0	0.324648
8	0	3	0	-1.908136	3	1	-0.257546
9	1	3	1	-0.185937	0	2	-0.445544
10	1	2	1	-1.081480	1	0	-0.087435
11	1	3	1	-1.770360	1	1	-0.340278
12	1	1	1	1.949591	0	0	-0.154013
13	0	3	0	-0.668153	0	0	-0.503850
14	0	3	0	0.640719	1	5	-0.064663
15	0	3	1	-1.081480	0	0	-0.507552
16	1	2	1	1.742927	0	0	-0.353515

```
coordonnees = acp.fit_transform(titanicStd.iloc[:,1:])
```

```
acp.n_components_
```

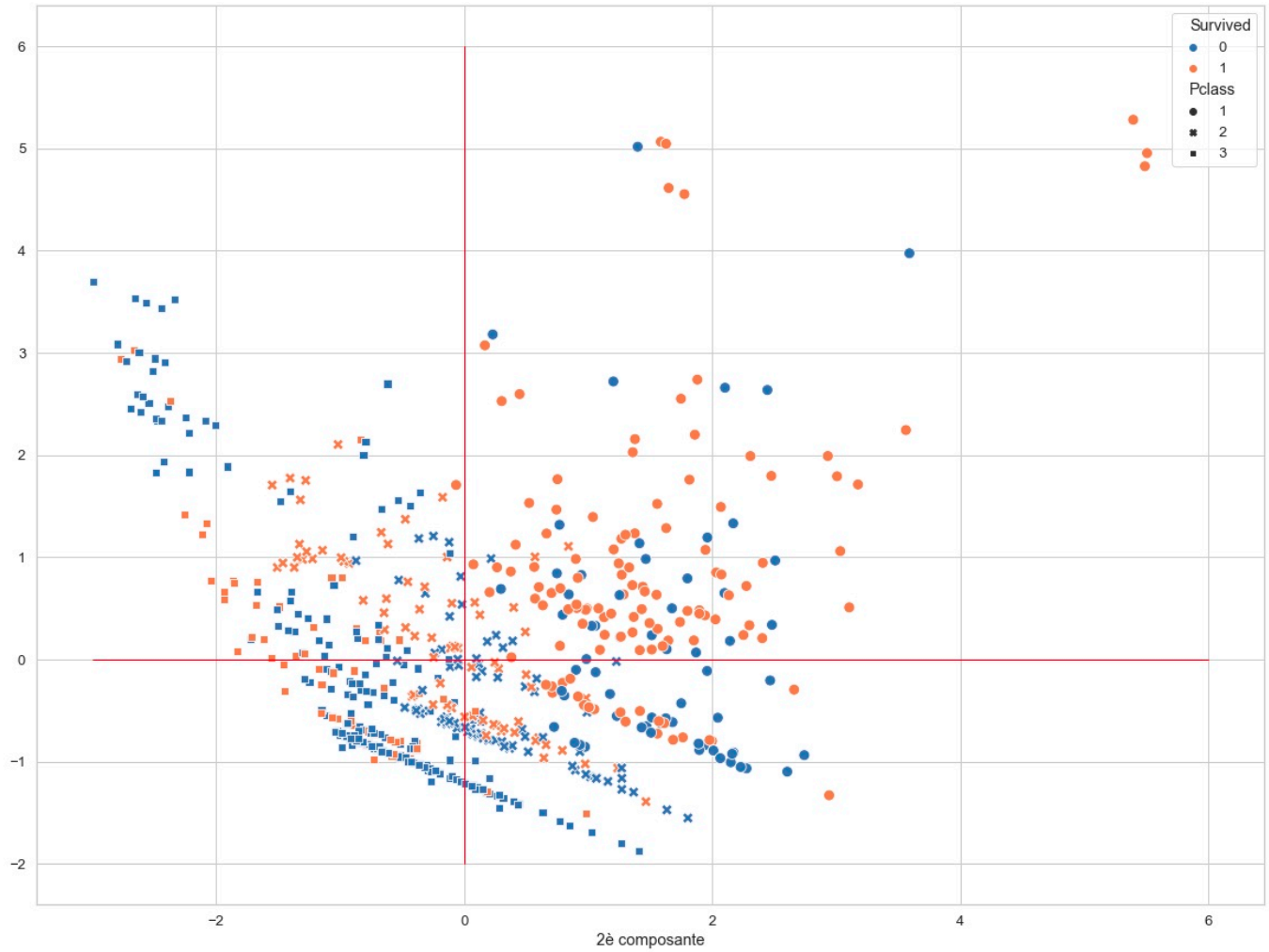
```
6
```

```
plt.plot(np.arange(1,acp.n_components_+1),acp.explained_variance_ratio_)
plt.title("Plot du % de variance expliquée")
plt.xticks(np.arange(1,acp.n_components_))
plt.ylabel("% de variance")
plt.xlabel("Numéro du facteur")
plt.show()
```



```
plt.figure(figsize=(20,15))
plt.plot([-limite+3,limite],[0,0],color='red',linestyle='-',linewidth=1)
plt.plot([0,0],[-limite+4,limite],color='red',linestyle='-',linewidth=1)
plt.xlabel("1ère composante")
plt.xlabel("2è composante")
sns.scatterplot(data=coordonnees, x=coordonnees[:,0], y=coordonnees[:,1], hue=titanicSansNA.Survived, style=titanicSansNA.Pclass, s=100)
```

```
<AxesSubplot:xlabel='2è composante'>
```



```
colnames = list(titanicSansNA.columns[1:])
prinComp_df = pd.DataFrame({ 'Feature':colnames,'PC1':acp.components_[0],'P
prinComp_df
```

	Feature	PC1	PC2	PC3	PC4	PC5
0	Pclass	-0.536399	-0.210316	-0.119329	0.138448	0.75
1	Sex	0.017707	0.128583	0.016152	0.149249	-0.27
2	Age	0.613889	-0.290566	-0.687855	-0.062978	0.21
3	SibSp	-0.280339	0.545620	-0.412939	-0.671782	-0.00
4	Parch	-0.136306	0.474719	-0.452647	0.708419	-0.10
5	Fare	0.487785	0.575974	0.370063	0.037963	0.53

```
fig, axes = plt.subplots(figsize=(12,12))
axes.set_xlim(-1,1)
axes.set_ylim(-1,1)
plt.scatter(prinComp_df.PC1, prinComp_df.PC2)
for i, txt in enumerate(prinComp_df.Feature):
    plt.annotate(txt,(prinComp_df.PC1[i],prinComp_df.PC2[i]),fontsize=12,  horizontalalignment='right', fontweight='bold')
    plt.arrow(0,0,prinComp_df.PC1[i]*0.9,prinComp_df.PC2[i]*0.9, linestyle=':', capstyle="round", width=0.003, head_width=0.03)
plt.plot([-1,1],[0,0],color='silver',linestyle='-',linewidth=1)
plt.plot([0,0],[-1,1],color='silver',linestyle='-',linewidth=1)
cercle = plt.Circle((0,0),1,color='blue',fill=False)
axes.add_artist(cercle)
plt.xlabel('Principal Component 1',fontsize=12, fontweight='bold')
plt.ylabel('Principal Component 2',fontsize=12, fontweight='bold')
plt.show()
```