

# enedis\_cartes

April 7, 2021

## 1 Tracer une carte en Python

Le notebook propose plusieurs façons de tracer une carte en Python.

Il y a principalement trois façons de tracer une carte. La première est statique avec des modules comme [basemap](#) (*plus trop maintenu*) ou [cartopy](#) qui sont des surcouches de [matplotlib](#). Le second moyen est une carte animée ou non dans un notebook avec des modules comme [pygal](#), [plotly](#). La dernière consiste à insérer des éléments sur une carte en ligne telle que [OpenStreetMap](#) et le module [folium](#) ou [ipyleaflet](#).

Il y a souvent trois problèmes avec les cartes. Le premier sont avec les coordonnées. Les plus utilisées sont les coordonnées [longitude / latitude](#). Le problème est chaque pays a son propre système adapté à sa position géographique. Il faut souvent convertir (voir [lambert93\\_to\\_WGPS](#), [pyproj](#)). Le second problème est l'ajout de repères géographiques (frontières, fleuves, ...). Certains modules contiennent certaines informations, souvent pour les Etats-Unis. Mais souvent, il faut récupérer ces informations sur les sites open data de chaque pays : [départements français](#). La troisième difficulté est qu'on veut tracer des cartes très chargées et cela prend un temps fou.

```
[1]: from jupyterlab import add_notebook_menu
      add_notebook_menu()
```

```
[1]: <IPython.core.display.HTML object>
```

```
[2]: %matplotlib inline
```

### 1.1 données

```
[3]: from papierstat.datasets import load_enedis_dataset
      df = load_enedis_dataset()
      df.head(n=2).T
```

```
[3]:      0  \
      Année
      2016
      Nom commune
      Ponteilla
      Code commune
      66145
      Nom EPCI
      (Pmcu)
      Code EPCI
      200027183
      Type EPCI
      CU
      CU Perpignan Méditerranée
```

Nom département	Pyrénées-
Orientales	
Code département	
66	
Nom région	
Occitanie	
Code région	
76	
Domaine de tension	BT >
36 kVA	
Nb sites Photovoltaïque Enedis	
73	
Energie produite annuelle Photovoltaïque Enedis...	
10728.6	
Nb sites Eolien Enedis	
0	
Energie produite annuelle Eolien Enedis (MWh)	
0	
Nb sites Hydraulique Enedis	
0	
Energie produite annuelle Hydraulique Enedis (MWh)	
0	
Nb sites Bio Energie Enedis	
0	
Energie produite annuelle Bio Energie Enedis (MWh)	
0	
Nb sites Cogénération Enedis	
0	
Energie produite annuelle Cogénération Enedis (...)	
0	
Nb sites Autres filières Enedis	
0	
Energie produite annuelle Autres filières Enedi...	
0	
Geo Point 2D	42.6323626522,
2.82631103755	
long	
2.82631	
lat	
42.6324	
	1
Année	2016
Nom commune	Varreddes
Code commune	77483
Nom EPCI	CA Pays de Meaux
Code EPCI	247700628
Type EPCI	CA
Nom département	Seine-et-Marne
Code département	77
Nom région	Île-de-France
Code région	11
Domaine de tension	BT <= 36 kVA

Nb sites Photovoltaïque Enedis	10
Energie produite annuelle Photovoltaïque Enedis...	21.4168
Nb sites Eolien Enedis	0
Energie produite annuelle Eolien Enedis (MWh)	0
Nb sites Hydraulique Enedis	0
Energie produite annuelle Hydraulique Enedis (MWh)	0
Nb sites Bio Energie Enedis	0
Energie produite annuelle Bio Energie Enedis (MWh)	0
Nb sites Cogénération Enedis	0
Energie produite annuelle Cogénération Enedis (...)	0
Nb sites Autres filières Enedis	0
Energie produite annuelle Autres filières Enedi...	0
Geo Point 2D	49.0059497861, 2.92725176893
long	2.92725
lat	49.0059

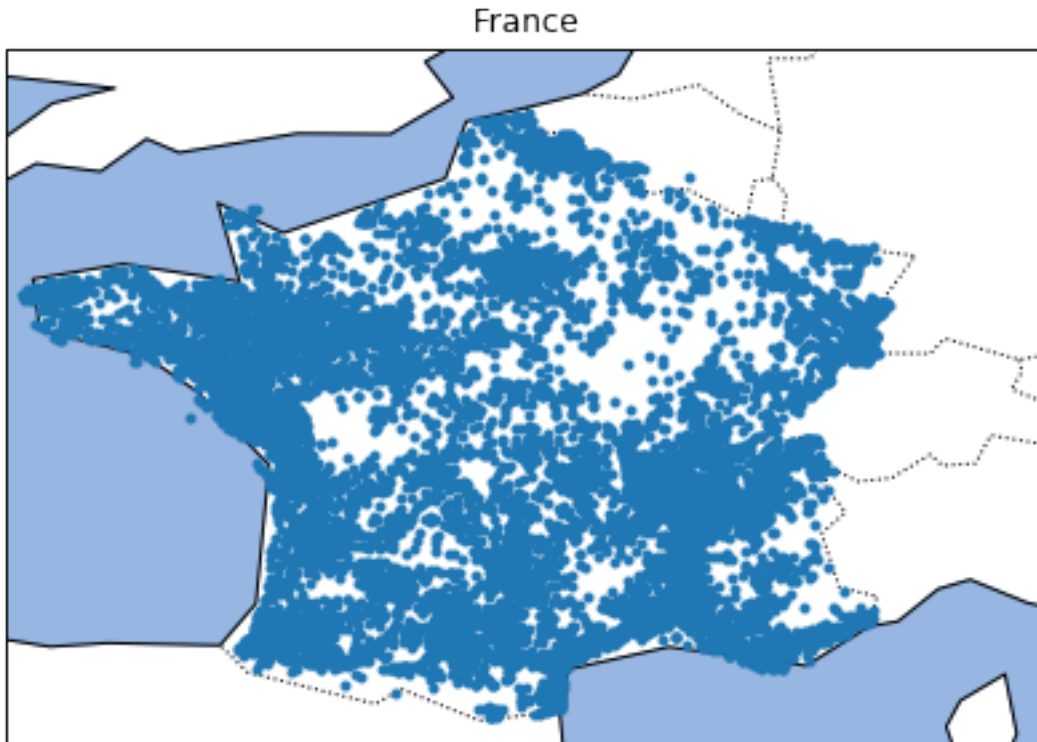
## 1.2 cartopy

`basemap` est l'ancêtre des modules de tracé de cartes sous Python mais il n'est plus vraiment maintenu. Il faut utiliser `cartopy`. Contrairement à `basemap`, `cartopy` n'installe pas toutes les données dont il a besoin mais télécharge celle dont il a besoin pour tracer une carte. La `projection` indique comment la surface de la terre, sphérique, sera projetée dans le plan. Ensuite, il faut un système de coordonnées pour localiser un point sur la surface. Le plus utilisée est `WGS_84` ou longitude, latitude. En France, l'INSEE utilise aussi le système `Lambert 93` ou `EPSG 2154`. Source : [Introduction à la manipulation de données cartographiques](#). Tout n'est pas parfait dans Cartopy comme ce problème [Create Cartopy projection from pyproj.Proj](#).

```
[4]: import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(1, 1, 1, projection=ccrs.PlateCarree())
ax.set_extent([-5, 10, 42, 52])

ax.add_feature(cfeature.OCEAN)
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')
ax.plot(df.long, df.lat, '.')
ax.set_title('France');
```



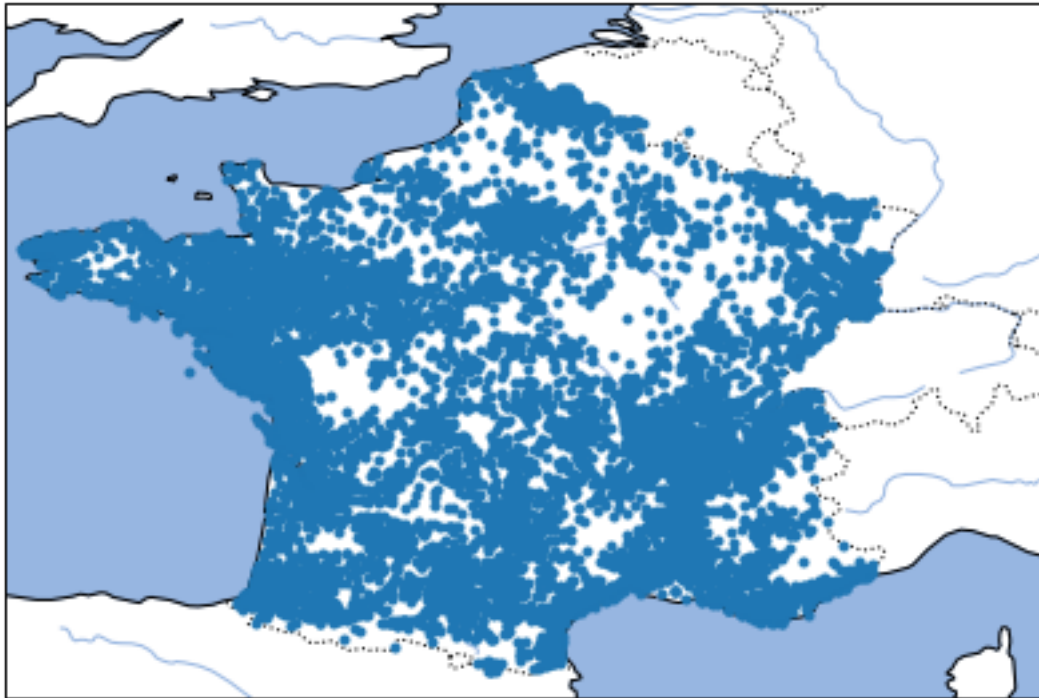
On peut obtenir une carte un peu plus détaillée mais le module `cartopy` télécharge des données pour ce faire. Cela se fait avec l'instruction `with_scale` qui propose trois résolutions :10m, 50m, 110m.

```
[5]: import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(1, 1, 1, projection=ccrs.PlateCarree())
ax.set_extent([-5, 10, 42, 52])

ax.add_feature(cfeature.OCEAN.with_scale('50m'))
ax.add_feature(cfeature.COASTLINE.with_scale('50m'))
ax.add_feature(cfeature.RIVERS.with_scale('50m'))
ax.add_feature(cfeature.BORDERS.with_scale('50m'), linestyle=':')
ax.plot(df.long, df.lat, '.')
ax.set_title('France');
```

## France



On ajoute un système de coordonnées français particulièrement intéressant pour la France. On convertit d'abord longitude, latitude en Lambert 93.

```
[6]: from pyproj import Proj, transform
p1 = Proj(init='epsg:4326') # longitude / latitude
p2 = Proj(init='epsg:2154') # Lambert 93
transform(p1, p2, -5, 42)
```

```
[6]: (36954.44139527541, 6133555.145153085)
```

```
[7]: transform(p1, p2, 10, 52)
```

```
[7]: (1181938.177574663, 7233428.2223392185)
```

On convertit toutes les coordonnées.

```
[8]: lamb_x, lamb_y = transform(p1, p2, df.long.values, df.lat.values)
```

Et on dessine deux cartes, la première en longitude, latitude, la seconde en Lambert 93.

```
[9]: import cartopy.crs as ccrs
from cartopy.crs import CRS, Globe
import cartopy.feature as cfeature
import matplotlib.pyplot as plt

def parse_option_pyproj(s):
    r = s.strip('+').split('=')
    if len(r) == 2:
        if ',' in r[1]:
```

```

        return r[0], tuple(int(_) for _ in r[1].split(','))
    try:
        return r[0], float(r[1])
    except ValueError:
        return r[0], r[1]
else:
    return r[0], True

class MyCRS(CRS):
    def __init__(self, proj4_params, globe=None):
        super(MyCRS, self).__init__(proj4_params, globe or Globe())

# voir https://epsg.io/2154, cliquer sur proj.4
proj4_params = "+proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=700000 " + \
               "+y_0=6600000 +ellps=GRS80 +towgs84=0,0,0,0,0,0 +units=m +no_defs"

# Ne sert à rien si ce n'est à vérifier que le format est correct.
import pyproj
lambert93 = pyproj.Proj(proj4_params)

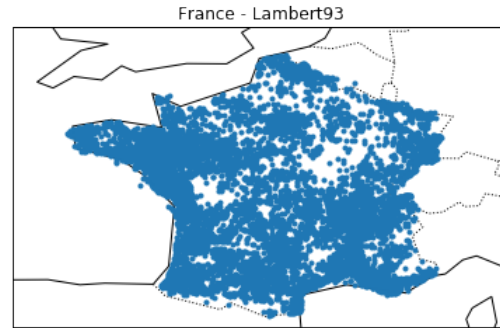
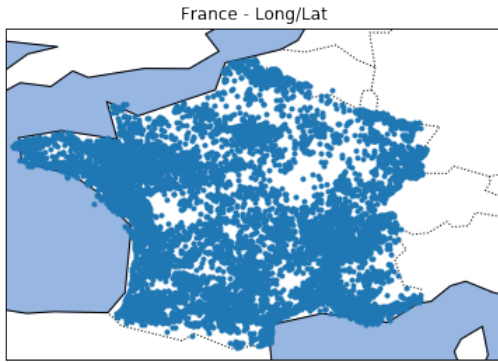
# Système de coordonnées de cartopy.
proj4_list = [(k, v) for k,v in map(parse_option_pyproj, proj4_params.split())]
crs_lambert93 = MyCRS(proj4_list, globe=None)

fig = plt.figure(figsize=(14,7))
ax = fig.add_subplot(1, 2, 1, projection=ccrs.PlateCarree())
ax.set_extent([-5, 10, 42, 52])

ax.add_feature(cfeature.OCEAN)
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')
ax.plot(df.long, df.lat, '.') #
ax.set_title('France - Long/Lat')

df = df.copy()

ax = fig.add_subplot(1, 2, 2, projection=ccrs.PlateCarree())
ax.set_extent([36954, 1181938, 6133555, 7233428], crs_lambert93)
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')
ax.plot(lamb_x, lamb_y, '.', transform=crs_lambert93) # ne pas oublier
↳transform=crs_lambert93
ax.set_title('France - Lambert93');
```



### 1.3 plotly, gmaps, bingmaps

Il faut s'authentifier. Voir [gmaps](#), [bingmaps](#), [plotly](#).

### 1.4 geopandas

[geopandas](#) est l'outil qui devient populaire. La partie cartes est accessible via l'API de [geopandas](#). Il n'inclut moins de données que *basemap*.

```
[10]: import geopandas as gpd
cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
cities.head()
```

```
[10]:
```

	name	geometry
0	Vatican City	POINT (12.45338654497177 41.90328217996012)
1	San Marino	POINT (12.44177015780014 43.936095834768)
2	Vaduz	POINT (9.516669472907267 47.13372377429357)
3	Luxembourg	POINT (6.130002806227083 49.61166037912108)
4	Palikir	POINT (158.1499743237623 6.916643696007725)

```
[11]: world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
world.head()
```

```
[11]:
```

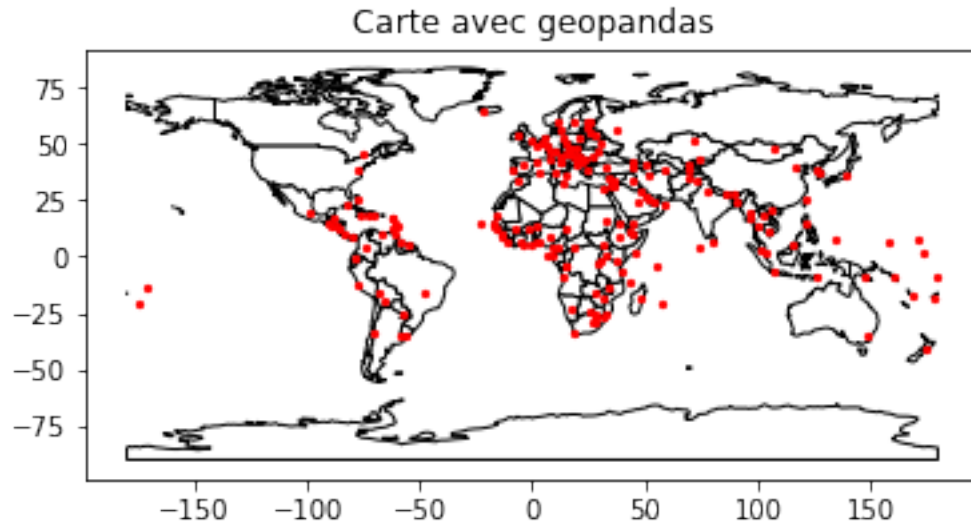
	pop_est	continent	name	iso_a3	gdp_md_est	\
0	28400000.0	Asia	Afghanistan	AFG	22270.0	
1	12799293.0	Africa	Angola	AGO	110300.0	
2	3639453.0	Europe	Albania	ALB	21810.0	
3	4798491.0	Asia	United Arab Emirates	ARE	184300.0	
4	40913584.0	South America	Argentina	ARG	573900.0	

```

geometry
0 POLYGON ((61.21081709172574 35.650072333330923,...
1 (POLYGON ((16.32652835456705 -5.87747039146621...
2 POLYGON ((20.59024743010491 41.85540416113361,...
3 POLYGON ((51.57951867046327 24.24549713795111,...
4 (POLYGON ((-65.50000000000003 -55.19999999999999...
```

```
[12]: import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 1, figsize=(6, 6))
```

```
ax.set_aspect('equal')
world.plot(ax=ax, color='white', edgecolor='black')
cities.plot(ax=ax, marker='o', color='red', markersize=5)
ax.set_title('Carte avec geopandas');
```



On restreint à l'Europe et pas trop loin de la France métropole.

```
[13]: from shapely.geometry import Polygon
europe = world[world.continent == "Europe"].copy()
europe['geometry'] = europe.geometry.intersection(Polygon([(-10,35), (50,35), (50,70),
↳ (-10, 70)]))
europe.head()
```

```
[13]:
```

	pop_est	continent	name	iso_a3	gdp_md_est	\
2	3639453.0	Europe	Albania	ALB	21810.0	
9	8210281.0	Europe	Austria	AUT	329500.0	
12	10414336.0	Europe	Belgium	BEL	389300.0	
16	7204687.0	Europe	Bulgaria	BGR	93750.0	
18	4613414.0	Europe	Bosnia and Herz.	BIH	29700.0	

```

geometry
2 POLYGON ((20.59024743010491 41.85540416113361,...
9 POLYGON ((16.97966678230404 48.12349701597631,...
12 POLYGON ((3.314971144228537 51.34578095153609,...
16 POLYGON ((22.65714969248299 44.23492300066128,...
18 POLYGON ((19.00548628101012 44.86023366960916,...
```

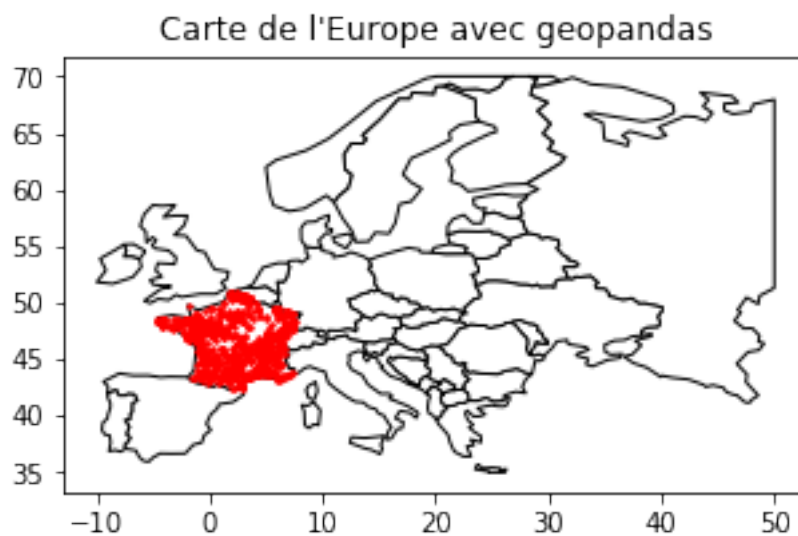
```
[14]: from shapely.geometry import Point
points = [Point(lon, lat) for ind, lat, lon in df[['lat', 'long']][:1000].itertuples()]
enedis = gpd.GeoDataFrame(data=dict(geometry=points))
enedis.head()
```



```
[14]: geometry
0 POINT (2.82631103755 42.6323626522)
1 POINT (2.92725176893 49.0059497861)
2 POINT (4.21389044246 44.4604648196)
3 POINT (0.974214866115 47.1204664673)
4 POINT (5.08531612205 48.6170558356)
```

```
[15]: fig, ax = plt.subplots(1, 1, figsize=(5, 4))
       europe.plot(ax=ax, color='white', edgecolor='black')
       enedis.plot(ax=ax, marker='o', color='red', markersize=2)
       ax.set_title("Carte de l'Europe avec geopandas");
```

```
[15]: Text(0.5,1,"Carte de l'Europe avec geopandas")
```



## 1.5 folium

```
[16]: import folium
       map_osm = folium.Map(location=[48.85, 2.34])

       for ind, lat, lon, com in df[['lat', 'long', 'Nom commune']][:50].itertuples():
           map_osm.add_child(folium.RegularPolygonMarker(location=[lat,lon], popup=com,
                                                         fill_color='#132b5e', radius=5))

       map_osm
```

```
[16]: <folium.folium.Map at 0x1e51f222c18>
```

## 1.6 cartopy avec les données d'OpenStreetMap

On peut choisir également d'inclure ces détails dans une image fixe si l'image va dans un rapport écrit. On utilise les données d'[OpenStreetMap](#) avec un certain [niveau de détail](#).

```
[17]: import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.pyplot as plt
from cartopy.io.ing_tiles import OSM

fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(1, 1, 1, projection=ccrs.PlateCarree())
ax.set_extent([-5, 10, 42, 52])

imagery = OSM()
ax.add_image(imagery, 5)
# plus c'est grand, plus c'est précis, plus ça prend du temps

ax.plot(df.long[:5], df.lat[:5], '.')
ax.set_title('France');
```



[18]: