J'ai choisi un jeu de données sur un parc de 4 éoliennes en open data. Ce jeu de données se nomme: "La haute borne" (https://opendata-renewables.engie.com/explore/dataset/data_description/table/?q=kw&dataChart=eyJxdWVyaWVzIjpbeyJjb25maWciOnsiZGF0YXNldCI6ImRhdGFfZGVzY3Jp du nom du parc éolien, situé dans la Meuse. L'analyse debute au 31 decembre 2016 et se termine au 12 janvier 2018.

In [1]:
```
import pandas as pd
import time
import datetime
import matplotlib.pyplot as plt
```

In [2]:
```
pwd()
```

Out[2]:
```
'/home/administrateur'
```

In [3]:
```
df = pd.read_csv('./STA002-2019/data/la-haute-borne-data-2017-2020.csv', sep=',')
```

In [4]:
```
df.shape
```

Out[4]:
```
(217588, 138)
```

Le data frame comporte 138 colonnes et 217588 lignes, nous allons donc selectionner que les colonnes nous interessant pour tracer la courbe de puissance d'une eolienne

* Date
* Nom eolienne
* vitesse vent
* puissance active (kwt)

Les colonnes sélectionnées sont:

- 'Wind_turbine_name',
- 'Date_time',
- 'P_avg',
- 'Ws1_avg',
- 'Ws2_avg',
- 'Ws_avg',

In [6]:
```
df1 = df.loc[:,['Wind_turbine_name','Date_time','P_avg','Ws1_avg','Ws2_avg','Ws_avg']]
```

In [7]:
```
df1.shape
```

Out[7]:
```
(217588, 6)
```

Traitement des valeurs NA

Y a t-il des valeurs NA? Remplacons les valeurs NA par 0

In [8]:
```
df1.isna().sum()
```

Out[8]:
```
Wind_turbine_name       0
Date_time               0
P_avg                1658
Ws1_avg              2937
Ws2_avg              2901
Ws_avg               1658
dtype: int64
```

In [9]:
```
df1=df1.fillna(0)
```

In [10]:
```
df1.dtypes
```

Out[10]:
```
Wind_turbine_name     object
Date_time             object
P_avg                float64
Ws1_avg              float64
Ws2_avg              float64
Ws_avg               float64
dtype: object
```

Le data frame ne comporte plus que 6 colonnes, nous creeons les colonnes Date, Hour et Minute

In [11]:
```
df1['Date_time'] = pd.to_datetime(df1['Date_time'])
df1['Date']=df1['Date_time'].dt.date
df1['Date']=pd.to_datetime(df1['Date'])
df1['Time']=df1['Date_time'].dt.time
df1['Hour']=df1['Date_time'].dt.hour
df1['minute']=df1['Date_time'].dt.minute

df1['Wind_turbine_name']=df1['Wind_turbine_name'].astype('category')
```

In [12]:
```
df1.dtypes
```

Out[12]:
```
Wind_turbine_name          category
Date_time            datetime64[ns]
P_avg                       float64
Ws1_avg                     float64
Ws2_avg                     float64
Ws_avg                      float64
Date                 datetime64[ns]
Time                         object
Hour                          int64
minute                        int64
dtype: object
```

In [13]:
```
df1.head()
```

Out[13]:

	Wind_turbine_name	Date_time	P_avg	Ws1_avg	Ws2_avg	Ws_avg	Date	Time	Hour	minute
0	R80711	2017-01-26 10:30:00	340.320010	5.54	6.14	5.84	2017-01-26	10:30:00	10	30
1	R80711	2017-02-21 13:00:00	790.669980	7.97	7.56	7.76	2017-02-21	13:00:00	13	0
2	R80711	2017-02-07 23:40:00	41.389999	3.97	4.07	4.02	2017-02-07	23:40:00	23	40
3	R80711	2017-02-08 00:40:00	-0.200000	2.69	3.00	2.84	2017-02-08	00:40:00	0	40
4	R80711	2017-02-21 18:50:00	522.719970	7.04	6.59	6.82	2017-02-21	18:50:00	18	50

In [14]:
```
df1.shape
```

Out[14]:
```
(217588, 10)
```

La methode min et max sur la series Date_time permet d'identifier l'intervalle de temps.

In [15]:
```
df1.Date_time.min()
```

Out[15]:
```
Timestamp('2016-12-31 23:00:00')
```

In [16]:
```
df1.Date_time.max()
```

Out[16]:
```
Timestamp('2018-01-12 23:00:00')
```

Contrôle de l'intervalle de temps

Debut de l'intervalle

In [17]:
```
df1.sort_values(by = 'Date_time', ascending= True).head()
```

Out[17]:

	Wind_turbine_name	Date_time	P_avg	Ws1_avg	Ws2_avg	Ws_avg	Date	Time	Hour	minute
164504	R80790	2016-12-31 23:00:00	65.349998	4.28	4.58	4.43	2016-12-31	23:00:00	23	0
7325	R80736	2016-12-31 23:00:00	21.969999	3.79	4.43	4.11	2016-12-31	23:00:00	23	0
158057	R80721	2016-12-31 23:00:00	-5.800000	3.13	3.34	3.23	2016-12-31	23:00:00	23	0
880	R80711	2016-12-31 23:00:00	76.489998	4.62	4.51	4.56	2016-12-31	23:00:00	23	0
883	R80711	2016-12-31 23:10:00	82.330002	5.10	4.88	4.99	2016-12-31	23:10:00	23	10

Fin de l'intervalle

In [18]:
```
df1.sort_values(by = 'Date_time', ascending= False).head()
```

Out[18]:

	Wind_turbine_name	Date_time	P_avg	Ws1_avg	Ws2_avg	Ws_avg	Date	Time	Hour	minute
217457	R80790	2018-01-12 23:00:00	0.00	2.55	2.63	2.59	2018-01-12	23:00:00	23	0
203945	R80721	2018-01-12 23:00:00	0.00	1.61	2.08	1.85	2018-01-12	23:00:00	23	0
206459	R80736	2018-01-12 23:00:00	0.00	2.40	2.61	2.50	2018-01-12	23:00:00	23	0
204887	R80711	2018-01-12 23:00:00	24.73	3.38	3.66	3.52	2018-01-12	23:00:00	23	0
217424	R80790	2018-01-12 22:50:00	0.00	2.27	2.84	2.56	2018-01-12	22:50:00	22	50

Il n'y a pas de ligne vide

In [19]:
```
df1= df1.dropna(how='all', axis =0)
```

In [20]:
```
df1.shape
```

Out[20]: (217588, 10)

Traitement des doublons. Y-a-t-il des doublons dans le dataset de la Haute Borne?

In [21]:
```
df1.groupby(['Date_time','Wind_turbine_name','P_avg','Ws_avg','Ws1_avg','Ws2_avg'])['Wind_turbine_name'].value
_counts().loc[lambda x: x>1]
```

Out[21]:

Date_time	Wind_turbine_name	P_avg	Ws_avg	Ws1_avg	Ws2_avg	Wind_turbine_name	
2017-06-09 22:30:00	R80721	0.0	0.0	0.0	0.0	R80721	2
2017-06-10 14:40:00	R80721	0.0	0.0	0.0	0.0	R80721	2
2017-06-10 16:00:00	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 16:10:00	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 16:20:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
2017-06-10 16:30:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
2017-06-10 16:40:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 16:50:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 17:00:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 17:10:00	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 17:20:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 17:30:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 17:40:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
							..
2017-06-10 18:20:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 18:30:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 18:40:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 18:50:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:00:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:10:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:20:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:30:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:40:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2
2017-06-10 19:50:00	R80711	0.0	0.0	0.0	0.0	R80711	2
	R80721	0.0	0.0	0.0	0.0	R80721	2
	R80736	0.0	0.0	0.0	0.0	R80736	2

Name: Wind_turbine_name, Length: 69, dtype: int64

Supprimons les doublons

In [22]:
```
df2=df1.drop_duplicates()
```

In [23]:
```
str(df1.shape[0]-df2.shape[0])+" lignes en double ont été supprimées"
```

Out[23]: '69 lignes en double ont été supprimées'

Y a t il encore des lignes en double?

In [24]:
```
df2.groupby(['Date_time','Wind_turbine_name','P_avg','Ws_avg'])['Wind_turbine_name'].value_counts().loc[lambda
x: x>1]
```

Out[24]: Series([], Name: Wind_turbine_name, dtype: int64)

=> Il n y a donc plus de doublon

Les eoliennes suivantes ont ete identifées dans le data set:

In [25]:
```
df2['Wind_turbine_name'].value_counts()
```

Out[25]:
```
R80711    54412
R80736    54411
R80721    54407
R80790    54289
Name: Wind_turbine_name, dtype: int64
```

Pourquoi l'eolienne R80790 n'a t'elle pas le meme nombre d'observations que les eoliennes R80711, R80736, R80721? Nous allons essayer d'identifier l'ecart.

La fonction ci dessous, nous permet d'analyser le data frame pour 1 eolienne par Date:

In [26]:
```
def turbine(i):
    df3 = (df2.loc[(df2['Wind_turbine_name'] == i)])
    df3 = df3.loc[:,['Date']]
    df3=df3['Date'].value_counts(sort=True).head(15)
    return (df3)
```

In [27]:
```
turbine('R80721')
```

Out[27]:
```
2017-06-10    252
2017-06-09    154
2017-03-26    150
2017-05-26    144
2017-09-29    144
2017-11-20    144
2017-04-19    144
2017-11-07    144
2017-04-06    144
2017-10-25    144
2017-03-24    144
2017-10-12    144
2017-03-11    144
2017-02-26    144
2017-12-03    144
Name: Date, dtype: int64
```

Que s'est il passé le 10/06/2017 ?

On définit une fonction pour avoir une vision par jour et par heure

In [28]:
```
def jour(i,j):
    df_date=df2.loc[:,['Wind_turbine_name','Date','Time','Hour']]
    df_date = df_date.loc[(df_date['Date'] == i)&(df_date['Hour']==j)]
    df_date = df_date.groupby(['Date','Wind_turbine_name','Time']).count()

    return(df_date)
```

In [29]:
```
jour('2017-06-10',13)
```

Out[29]:

			Hour
Date	**Wind_turbine_name**	**Time**	
2017-06-10	**R80711**	**13:00:00**	2
		13:10:00	2
		13:20:00	2
		13:30:00	2
		13:40:00	2
		13:50:00	2
	R80721	**13:00:00**	2
		13:10:00	2
		13:20:00	2
		13:30:00	2
		13:40:00	2
		13:50:00	2
	R80736	**13:00:00**	2
		13:10:00	2
		13:20:00	2
		13:30:00	2
		13:40:00	2
		13:50:00	2
	R80790	**13:00:00**	1
		13:10:00	1
		13:20:00	1
		13:30:00	1
		13:40:00	1
		13:50:00	1

In [30]:
```
df_analyse = df2.loc[(df2['Date']=='2017-06-10 00:00:00')&(df2['Wind_turbine_name']=='R80721')]
df_analyse.sort_values(by='Time').head(15)
```

Out[30]:

	Wind_turbine_name	Date_time	P_avg	Ws1_avg	Ws2_avg	Ws_avg	Date	Time	Hour	minute
87423	R80721	2017-06-10 00:00:00	0.0	0.00	0.00	0.00	2017-06-10	00:00:00	0	0
175975	R80721	2017-06-10 00:00:00	0.0	2.65	2.45	2.55	2017-06-10	00:00:00	0	0
85716	R80721	2017-06-10 00:10:00	0.0	0.00	0.00	0.00	2017-06-10	00:10:00	0	10
48563	R80721	2017-06-10 00:10:00	0.0	2.88	2.62	2.75	2017-06-10	00:10:00	0	10
85717	R80721	2017-06-10 00:20:00	0.0	0.00	0.00	0.00	2017-06-10	00:20:00	0	20
175976	R80721	2017-06-10 00:20:00	0.0	3.07	2.85	2.96	2017-06-10	00:20:00	0	20
85718	R80721	2017-06-10 00:30:00	0.0	0.00	0.00	0.00	2017-06-10	00:30:00	0	30
48564	R80721	2017-06-10 00:30:00	0.0	2.48	2.43	2.46	2017-06-10	00:30:00	0	30
48565	R80721	2017-06-10 00:40:00	0.0	2.57	2.60	2.59	2017-06-10	00:40:00	0	40
85719	R80721	2017-06-10 00:40:00	0.0	0.00	0.00	0.00	2017-06-10	00:40:00	0	40
87424	R80721	2017-06-10 00:50:00	0.0	0.00	0.00	0.00	2017-06-10	00:50:00	0	50
175977	R80721	2017-06-10 00:50:00	0.0	2.74	2.75	2.74	2017-06-10	00:50:00	0	50
190582	R80721	2017-06-10 01:00:00	0.0	2.92	2.94	2.93	2017-06-10	01:00:00	1	0
87425	R80721	2017-06-10 01:00:00	0.0	0.00	0.00	0.00	2017-06-10	01:00:00	1	0
175978	R80721	2017-06-10 01:10:00	0.0	2.75	2.77	2.76	2017-06-10	01:10:00	1	10

Constat pour le 10/06/2017 et pour l'eolienne R80721: Il y a eu 2 prises de mesures par les anemometres à la meme heure, 1 observation est correcte, 1 observation affiche des mesures nulles, pour P_avg et Ws_avg.

```
* On va supprimer ces observations qui fausseront notre analyse graphique
```

In [31]:
```
df_nulle = df2[(df2['Ws_avg']==0)&(df2['P_avg']==0) ].index
```

In [32]:
```
df2.drop(df_nulle,inplace=True)
```

```
/home/administrateur/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:3697: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  errors=errors)
```

In [33]:
```
df2.shape
```

Out[33]:
```
(213365, 10)
```

Moyenne du vent sur le parc eolien de la haute borne

In [35]:
```
mean = round(df2["Ws_avg"].mean(skipna = True),2)
print(str(mean), "ms")
```

```
5.5 ms
```

In [36]:
```
df2['Wind_turbine_name'].value_counts()
```

Out[36]:
```
R80790    53567
R80721    53329
R80711    53296
R80736    53173
Name: Wind_turbine_name, dtype: int64
```

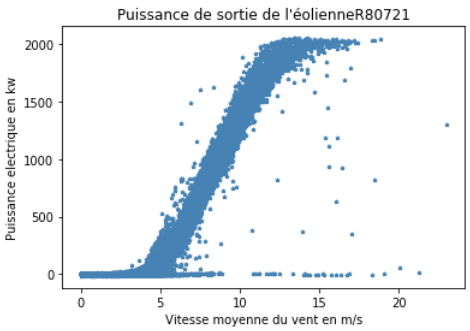
Pour la representation graphique nous effectuerons le nuage de points de la puissance d'une eolienne. Pour valider ce graphique, et pour une bonne comprehension des données, le tutoriel métier suivant a été suivi (https://eolienne.f4jr.org/eolienne_etude_theorique).

- Variable name : P, variable long name: active power, unit : kw
- variable name: ws, variable long name : wind speed, unit : m/s

Une fonction nous permettera d'afficher le graphique de l'eolienne sélectionnée

In [37]:
```
def graphique(i):
    df4 = (df2.loc[(df2['Wind_turbine_name'] == i)])
    plt.scatter(df4['Ws_avg'],df4['P_avg'],c = 'steelblue', marker ='*', s=10)
    plt.title('Puissance de sortie de l\'éolienne' + str(i))
    plt.ylabel('Puissance electrique en kw')
    plt.xlabel('Vitesse moyenne du vent en m/s')
    plt.show()
```

In [38]:
```
graphique('R80721')
```

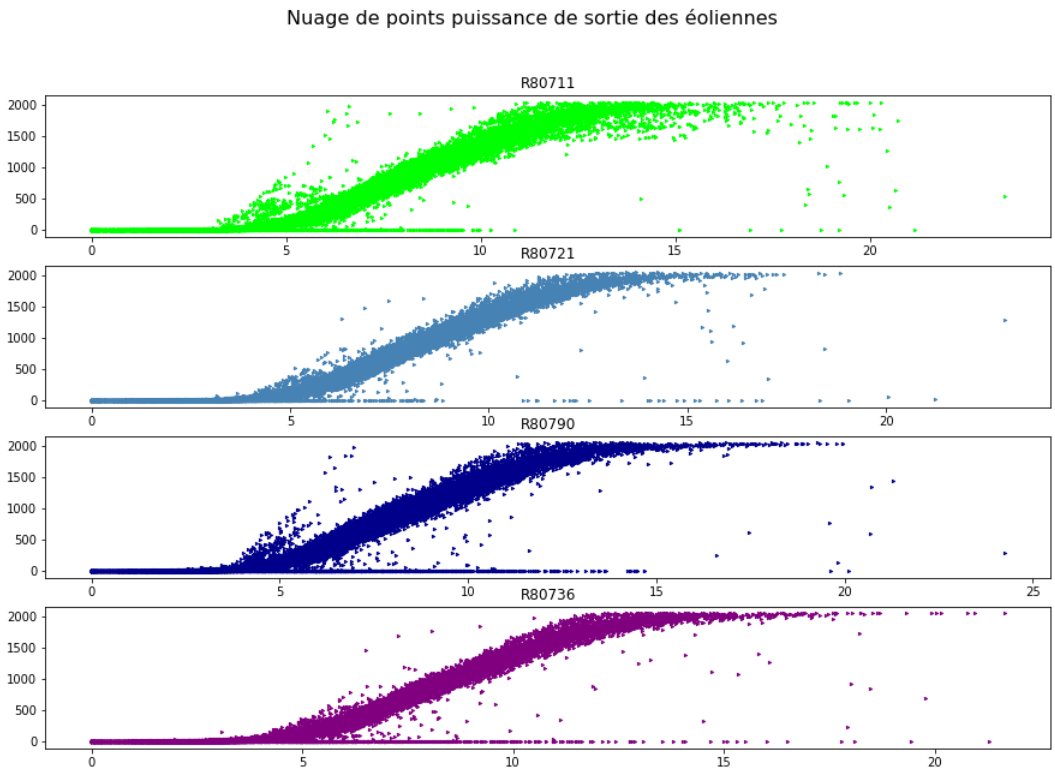


La methode subplot nous permet d'afficher les 4 éoliennes dans le meme graphique

In [39]:
```
a=df2.loc[(df2['Wind_turbine_name'] =='R80721')]['Ws_avg']
b=df2.loc[(df2['Wind_turbine_name'] =='R80721')]['P_avg']
c=df2.loc[(df2['Wind_turbine_name'] =='R80790')]['Ws_avg']
d=df2.loc[(df2['Wind_turbine_name'] =='R80790')]['P_avg']
e=df2.loc[(df2['Wind_turbine_name'] =='R80736')]['Ws_avg']
f=df2.loc[(df2['Wind_turbine_name'] =='R80736')]['P_avg']
g=df2.loc[(df2['Wind_turbine_name'] =='R80711')]['Ws_avg']
h=df2.loc[(df2['Wind_turbine_name'] =='R80711')]['P_avg']
```

In [40]:
```
fig, (ax1, ax2,ax3,ax4) = plt.subplots(nrows=4, ncols=1, figsize=(15, 10))
plt.suptitle('Nuage de points puissance de sortie des éoliennes', fontsize=16)
ax1.scatter(g, h, s=5, c='lime', marker=">")
ax1.set_title('R80711')
ax2.scatter(a, b, s=5, c='steelblue', marker=">")
ax2.set_title('R80721')
ax3.scatter(c, d, s=5, c='darkblue', marker=">")
ax3.set_title('R80790')
ax4.scatter(e, f, s=5, c='purple', marker=">")
ax4.set_title('R80736')
```

Out[40]:
```
Text(0.5, 1.0, 'R80736')
```



En reprenant le wiki éolienne et avec la lecture des differents graphiques:

- Juqu à la vitessse de démarrage environ 4 m/s soit 14 km/h : puissance de sortie nulle
- vitesse nominale environ 12 m/s soit 43,2 km/h
- Coupure des éoliennes environ 20 m/s soit 72 km/h: puissance de sortie nulle pour protection des éoliennes

In [ ]:

In [ ]: