

# Visualisation des données

Il existe différentes librairies qui permettent de visualiser les données contenues dans un dataframe. Par exemple Matplotlib ou Seaborn.

## Utilisation de Matplotlib

Il faut tout d'abord importer la librairie matplotlib

In [1]:

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

La librairie permet via la commande plot d'afficher différentes visualizations.

`df.plot(x=None, y=None, kind='line', ax=None, subplots=False, sharex=None, sharey=False, layout=None, figsize=None, use_index=True, title=None, grid=None, legend=True, style=None, logx=False, logy=False, loglog=False, xticks=None, yticks=None, xlim=None, ylim=None, rot=None, fontsize=None, colormap=None, table=False, yerr=None, xerr=None, secondary_y=False, sort_columns=False, **kwargs)`

Parmi les principales :

*kind* permet de spécifier le type de visualisation avec les valeurs dans :

'line' : line plot (default)

'bar' : vertical bar plot

'barh' : horizontal bar plot

'hist' : histogram

'box' : boxplot

'kde' : Kernel Density Estimation plot

'density' : same as 'kde'

'area' : area plot

'pie' : pie plot

'scatter' : scatter plot

'hexbin' : hexbin plot

*subplot* permet de spécifier qu'il s'agit d'une figure imbriquée dans une figure

*figsize* pour spécifier la taille de la figure via un tuple (width, height) en inches

*title* pour mettre un titre à la figure

Se reporter à la documentation officielle pour connaître la liste de toutes les options :

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html>  
(<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html>)

Quelques exemples à l'aide de la base IRIS

In [2]:

```
1 import pandas as pd
2 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
3 names = ['SepalLengthCm', 'SepalWidthCm',
4          'PetalLengthCm', 'PetalWidthCm', 'Species']
5
6 df = pd.read_csv(url, names=names)
7 # 5 premières lignes du fichier
8 df.head()
9
```

Out[2]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

## Affichage pour un attribut

Dans cette partie nous considérons chaque attribut indépendamment.

### Affichage des valeurs

Pour avoir une première idée des données manipulées

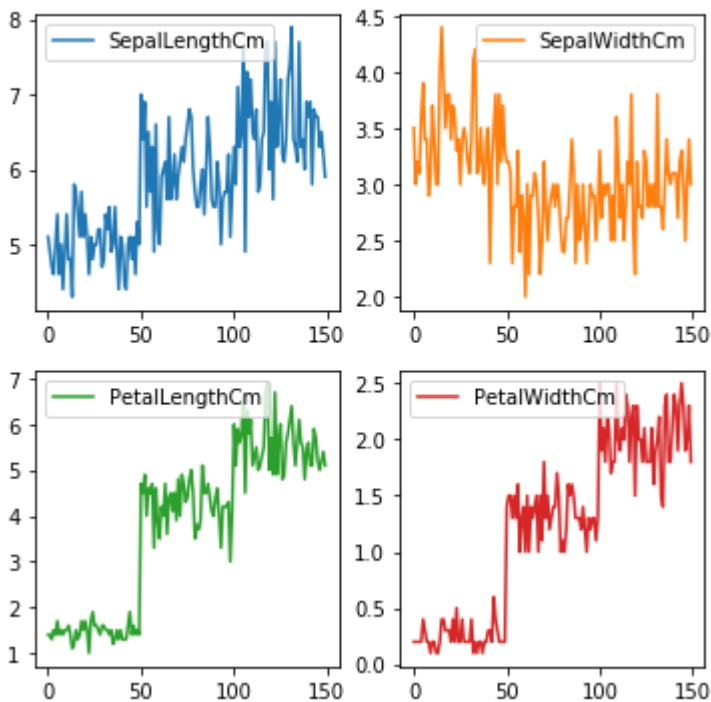
In [3]:

```

1  ▼ #Affichage des différentes valeurs
2  # remarque : ici nous considérons uniquement les valeurs numériques
3  # kind = 'line' pour avoir des histogrammes
4  # subplots= True pour afficher les différentes vues dans une seule figure
5  # layout = (4,4) permet d'afficher les 4 figures sur la même ligne.
6  # Par exemple layout=(2,2) affiche 2 figures sur une même ligne
7  # figsize = (6,6) indique la taille en inch.
8  # Attention en fonction du layout la taille ne sera pas prise en compte pour
9  # pouvoir insérer toutes les figures sur une ligne (layout = (4,4))
10 # sharex=False pour indiquer de mettre l'axe des x à chaque figure
11 # title='Valeurs des données' pour donner un titre à la figure
12 ▼ df.plot(kind='line', subplots=True,
13         layout=(2,2), figsize=(6,6),
14         sharex=False, title='Valeurs des données')
15 plt.show()

```

Valeurs des données



## Histogramme

Les histogrammes permettent de se faire une première idée de la distribution des données.

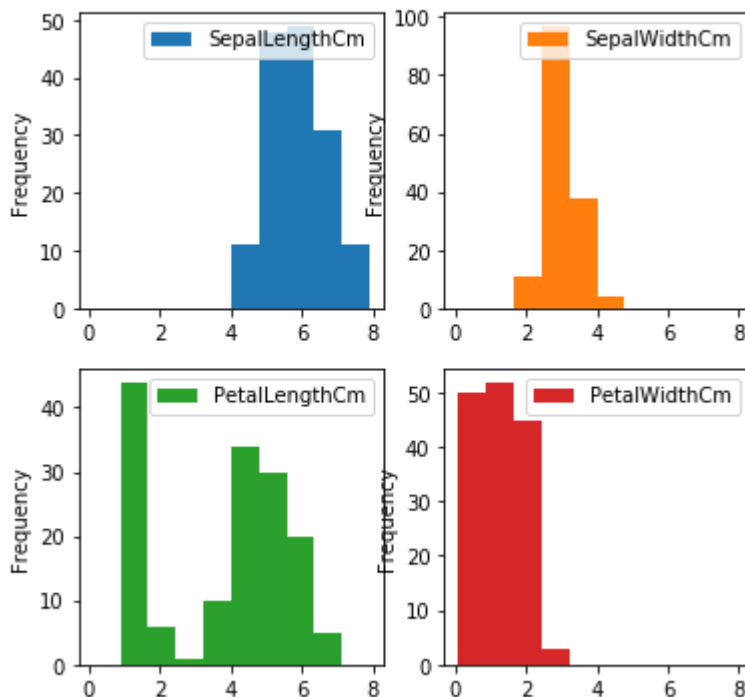
In [4]:

```

1  ▼ # Affichage des histogrammes des différentes valeurs
2  # kind = 'hist' pour avoir des histogrammes
3  # subplots= True pour afficher les différentes vues dans une seule figure
4  # layout = (4,4) permet d'afficher les 4 figures sur la même ligne.
5  # Par exemple layout=(2,2) affiche 2 figures sur une même ligne
6  # figsize = (6,6) indique la taille en inch.
7  # Attention en fonction du layout la taille ne sera pas prise en compte pour
8  # pouvoir insérer toutes les figures sur une ligne (layout = (4,4))
9  # sharex=False pour indiquer de mettre l'axe des x à chaque figure
10 # title='Histogramme' pour donner un titre à la figure
11 ▼ df.plot(kind='hist', subplots=True,
12           layout=(2,2), figsize=(6,6),
13           sharex=False, title='Histogramme')
14 plt.show()

```

Histogramme



### Affichages des densités

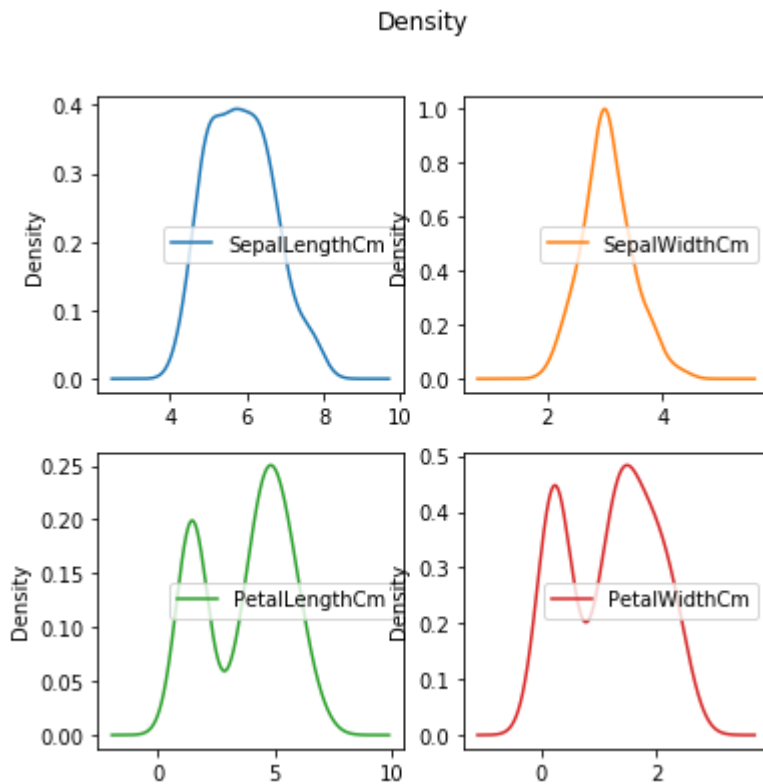
Pour avoir une vue de la densité sous la forme de courbes.

In [5]:

```

1  df.plot(kind='density',
2      subplots=True, layout=(2,2),
3      figsize=(6,6),sharex=False, title='Density')
4  plt.show()

```



### Affichage des boîtes à moustache

Les boîtes à moustache (boxplot) permettent d'avoir plus d'information sur la distribution de chaque attribut et notamment les outliers. Une boîte à moustache est un graphique simple composé d'un rectangle duquel deux droites sortent afin de représenter certains éléments des données :

La valeur centrale du graphique est la médiane (il existe autant de valeur supérieures qu'inférieures à cette valeur dans l'échantillon).

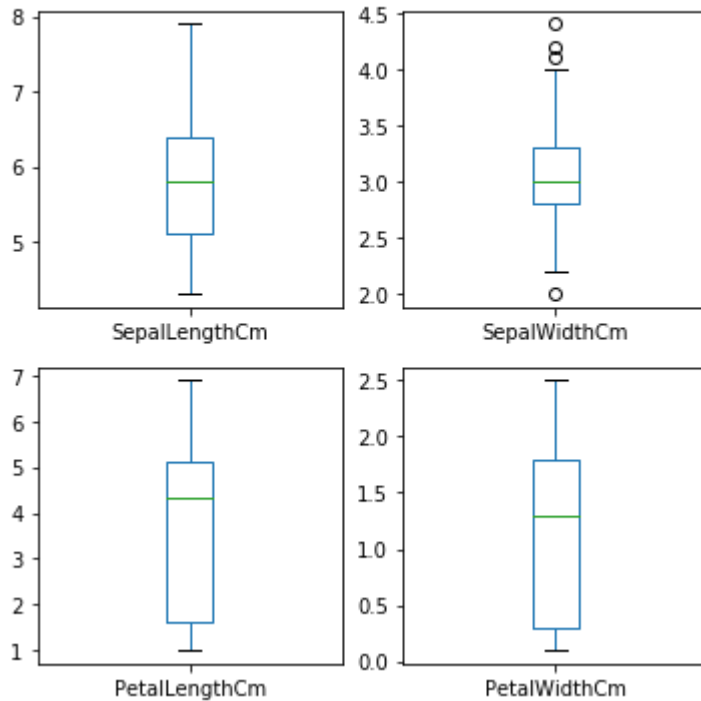
Les bords du rectangle sont les quartiles (Pour le bord inférieur, un quart des observations ont des valeurs plus petites et trois quart ont des valeurs plus grandes, le bord supérieur suit le même raisonnement).

Les extrémités des moustaches sont calculées en utilisant 1.5 fois l'espace interquartile (la distance entre le 1er et le 3ème quartile).

In [6]:

```
1 df.plot(kind='box', subplots=True,  
2         layout=(2,2), figsize=(6,6),  
3         sharex=False, title='Box-plot')  
4 plt.show()
```

Box-plot



### Diagramme circulaire

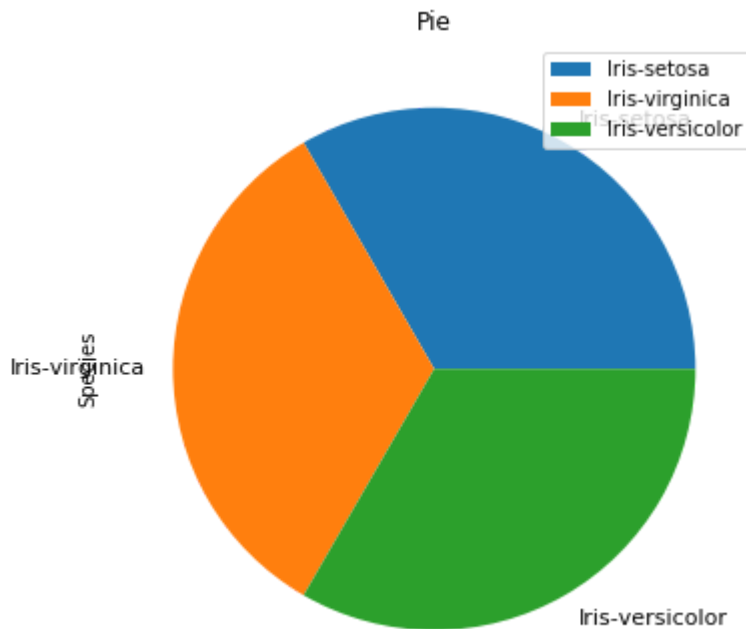
Les diagrammes circulaires (pie chart) permettent de connaître une répartition de valeurs.

In [7]:

```

1  # Species n'est pas une valeur numérique, ici on compte combien de valeurs (v
2  # fontsize pour changer la taille de la font
3  # legend=True pour afficher la légende
4  df['Species'].value_counts().plot(kind='pie',
5                                     figsize=(6,6),
6                                     title='Pie',
7                                     fontsize=11,
8                                     legend=True)
9  plt.show()

```



## Travailler avec plusieurs attributs

### Correlations et carte de chaleur

Il est possible via la fonction `correlation` de déterminer les corrélations qui existent entre les différents attributs. Les cartes de chaleur (heatmap) peuvent être utilisées pour mettre en évidence ces corrélations.

In [8]:

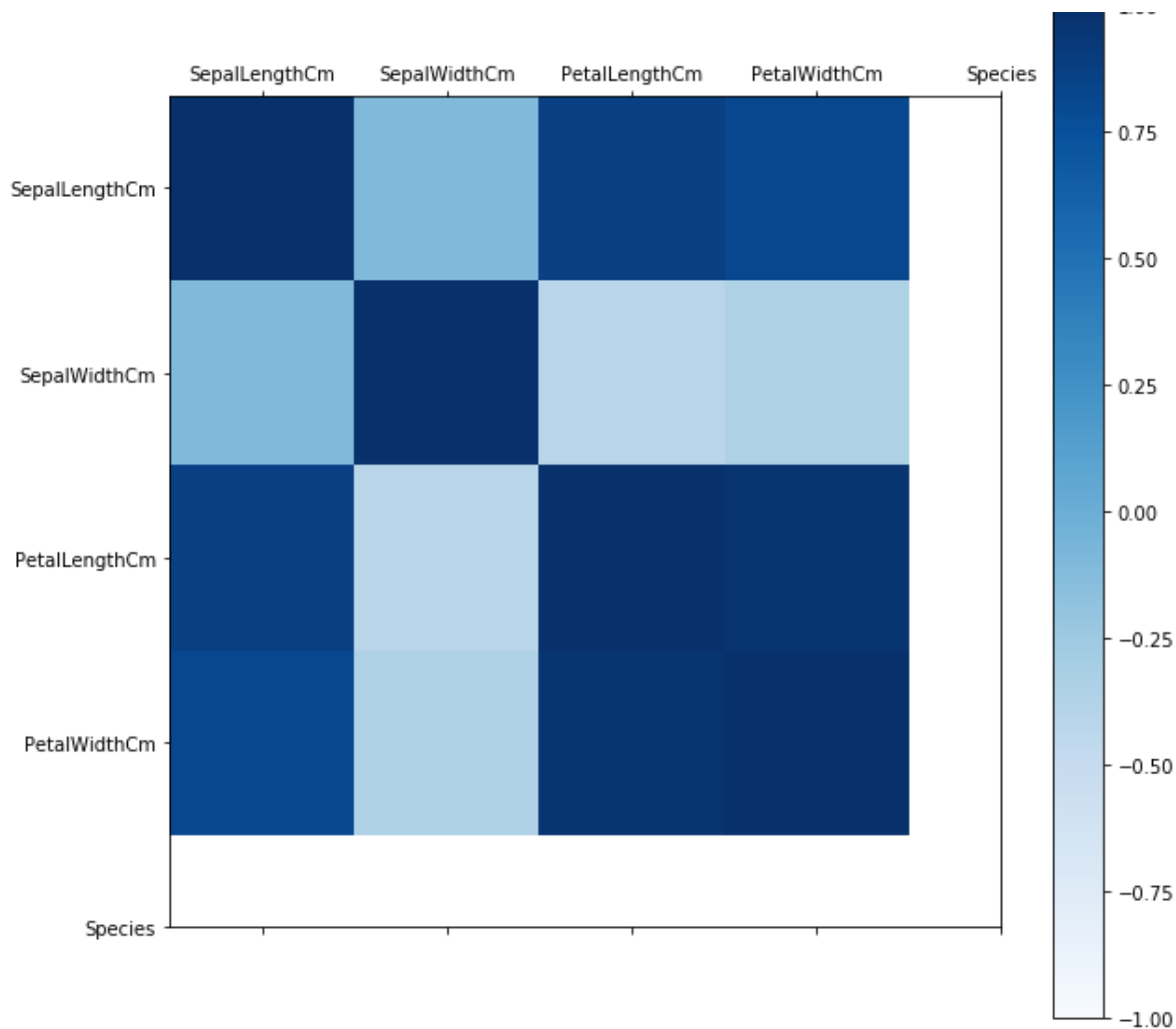
```

1  import numpy
2
3
4  correlations = df.corr()
5  # Affichage des valeurs de corrélation
6  print (correlations)
7
8  # Transformation des données de Species en données catégorielles
9  # pour pouvoir calculer aussi les corrélations avec les autres
10 from sklearn.preprocessing import LabelEncoder
11 le = LabelEncoder()
12 y=df['Species']
13 le.fit(y)
14 df['Species'] = le.transform(y)
15
16
17 # Affichage de la matrice de corrélation
18 fig = plt.figure(figsize=(10,10))
19 # Partie à droite pour montrer les variations de valeurs
20 ax = fig.add_subplot(111)
21 # Détermine les valeurs maximales et minimales à afficher et le thème Blues
22 cax = ax.matshow(correlations, vmin=-1,
23                  vmax=1,
24                  cmap=plt.cm.Blues)
25 # Récupération des couleurs en fonctions des valeurs
26 fig.colorbar(cax)
27 # Le tableau principal
28 # debute à 0, taille 5 (5 valeurs à afficher), largeur d'une colonne 1
29 ticks = numpy.arange(0,5,1)
30 # mise en place
31 ax.set_xticks(ticks)
32 ax.set_yticks(ticks)
33 ax.set_xticklabels(names)
34 ax.set_yticklabels(names)
35 plt.show()
36 # La diagonale est forcément la plus foncée

```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthC
m				
SepalLengthCm	1.000000	-0.109369	0.871754	0.81795
4				
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.35654
4				
PetalLengthCm	0.871754	-0.420516	1.000000	0.96275
7				
PetalWidthCm	0.817954	-0.356544	0.962757	1.00000
0				





### Nuages de points

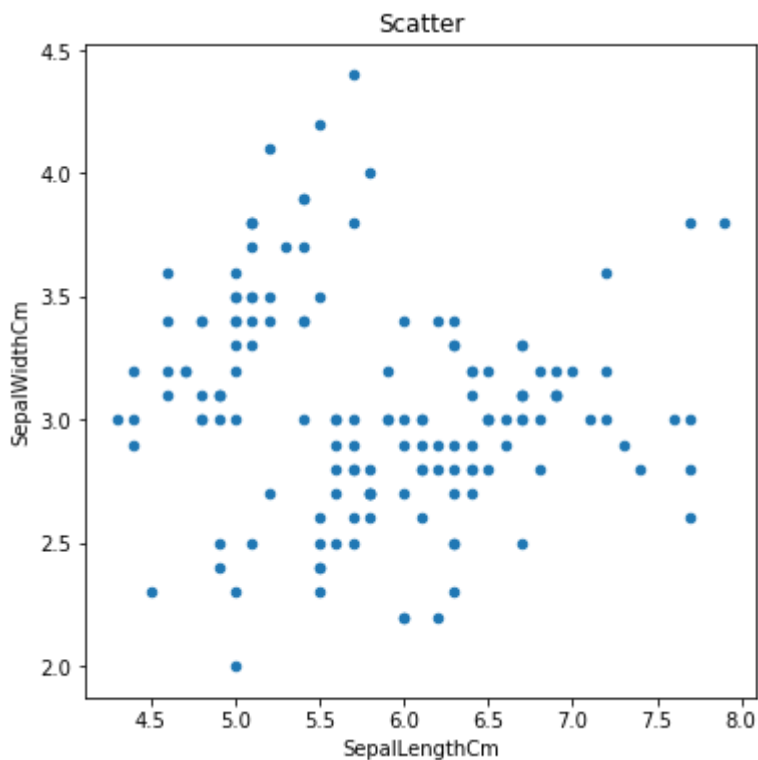
Ce nuage affiche les valeurs de deux variables. Elle est particulièrement utile pour savoir si deux valeurs sont fortement corrélés (ou inversement corrélées). Celles qui sont fortement corrélées peuvent être sans doute supprimées du jeu de données.

In [9]:

```

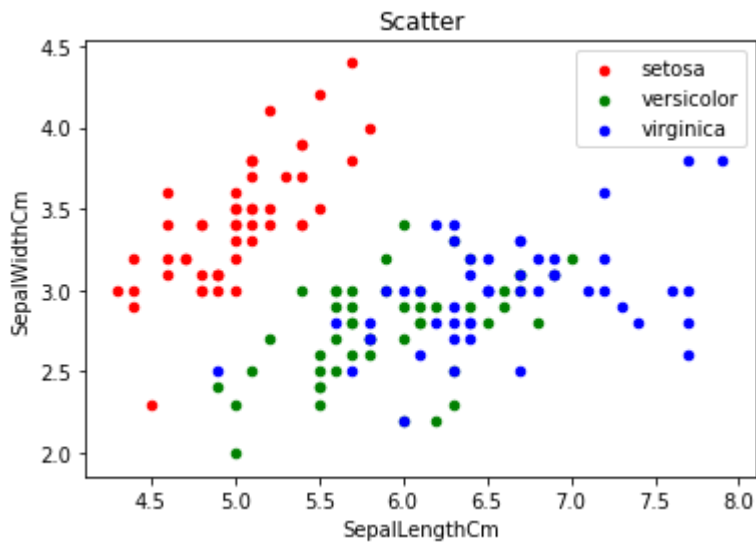
1  #df.plot(kind='scatter', subplots=True, layout=(2,2), figsize=(6,6),sharex=Fa
2  # Pour deux attributs il faut préciser le x et le y
3  url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.da
4  names = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'S
5
6  df = pd.read_csv(url, names=names)
7  df.plot(kind='scatter', x='SepalLengthCm',
8          y='SepalWidthCm',figsize=(6,6),
9          sharex=False, title='Scatter')
10 plt.show()
11
12 # Pour mettre en avant les différentes espèces
13 ax=df[df.Species=='Iris-setosa'].plot(kind='scatter',
14                                       x='SepalLengthCm',
15                                       y='SepalWidthCm',
16                                       color='red',
17                                       label='setosa')
18 df[df.Species=='Iris-versicolor'].plot(kind='scatter',
19                                         x='SepalLengthCm',
20                                         y='SepalWidthCm',
21                                         color='green',
22                                         label='versicolor',
23                                         ax=ax)
24 df[df.Species=='Iris-virginica'].plot(kind='scatter',
25                                       x='SepalLengthCm',
26                                       y='SepalWidthCm',
27                                       color='blue',
28                                       label='virginica',
29                                       ax=ax)
30 ax.set_title("Scatter")
31

```



Out[9]:

Text(0.5, 1.0, 'Scatter')



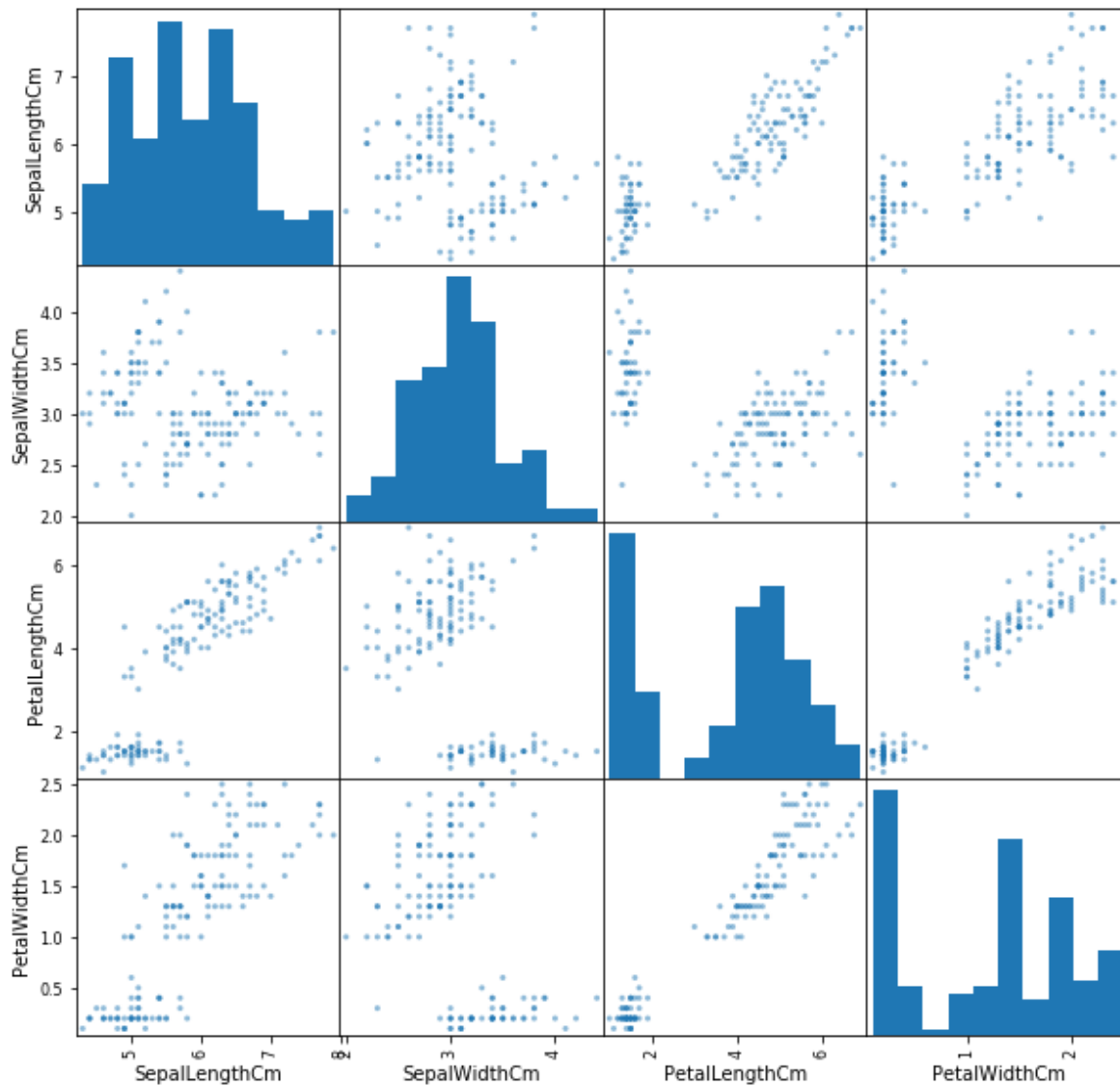
### Scatter plot matrix

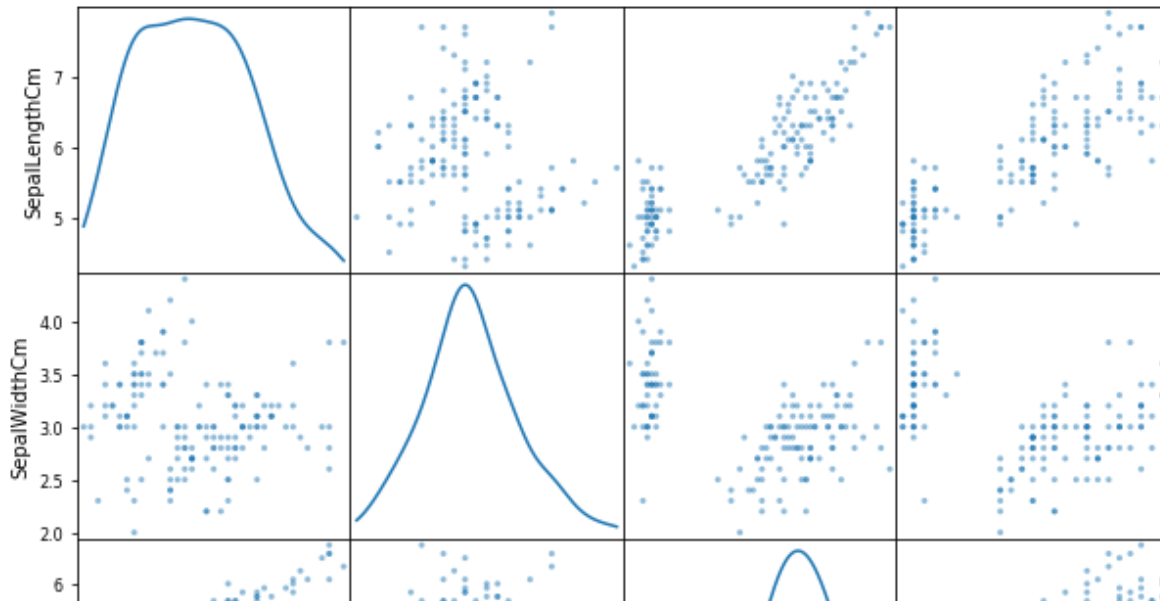
Cette matrice affiche toutes les valeurs en croisant les variables deux à deux.

Dans pandas il est possible d'utiliser la fonction `scatter_matrix` (voir [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.plotting.scatter\\_matrix.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.plotting.scatter_matrix.html)) qui offre toutes les fonctionnalités pour afficher le contenu de toutes les valeurs de la matrice.

In [10]:

```
1 from pandas.plotting import scatter_matrix
2 # Pour afficher toute la matrice
3 scatter_matrix(df,figsize=(10,10))
4 plt.show()
5
6 # Pour afficher la matrice mais avec les densités plutôt que des histogrammes
7 scatter_matrix(df,figsize=(10,10), diagonal='kde')
8 plt.show()
```





## Utilisation de Seaborn

Seaborn est une librairie plus récente développée au dessus de Matplotlib qui permet de pouvoir améliorer les visualisations. Une documentation complète est disponible : <https://seaborn.pydata.org> (<https://seaborn.pydata.org>).

Il faut importer la librairie seaborn

In [11]:

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
```

In [12]:

```
1 import pandas as pd
2 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.da
3 names = ['SepalLengthCm', 'SepalWidthCm',
4         'PetalLengthCm', 'PetalWidthCm', 'Species']
5
6 df = pd.read_csv(url, names=names)
7 # 5 premières lignes du fichier
8 df.head()
9
10
```

Out[12]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

## Affichage pour un attribut

## Aménagement pour un attribut

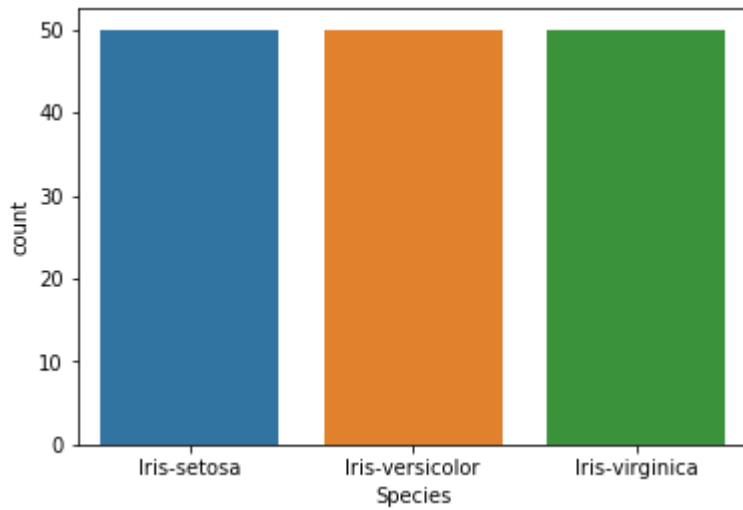
### Nombre de valeurs d'un attribut

In [13]:

```
1 sns.countplot(x='Species', data=df)
```

Out[13]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x116e6f6d8>



### Histogramme

In [14]:

```
1  # Pour avoir un fond blanc avec grille
2  sns.set(style="whitegrid")
3  # Avec une ligne verticale pour montrer chaque observation dans une distribut
4  sns.distplot(df["SepalLengthCm"])
5
```

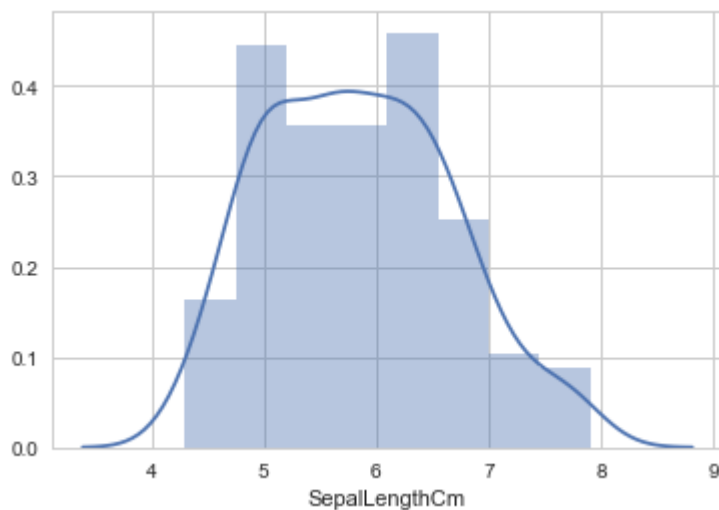
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/matplotlib/axes/\_axes.py:6521: MatplotlibDeprecationWarning:

The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.

```
    alternative="'density'", removal="3.1")
```

Out[14]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x116eb9d30>



In [15]:

```
1  ▾ # Sans ligne verticale  
2  sns.distplot(df["SepalLengthCm"], kde=False)
```

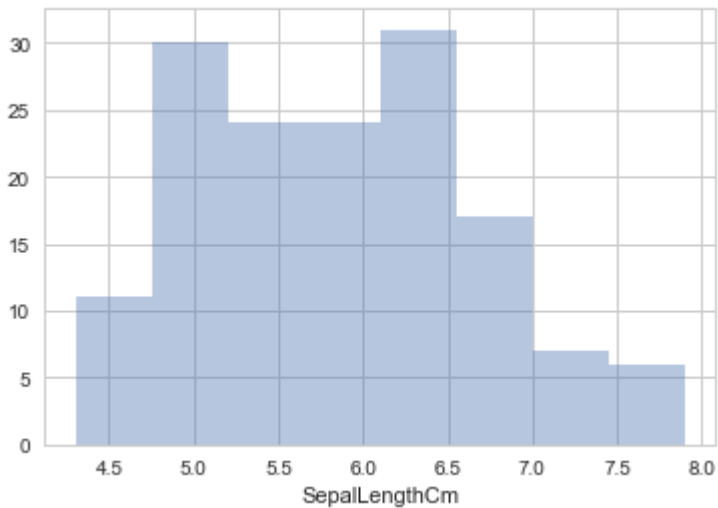
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/matplotlib/axes/\_axes.py:6521: MatplotlibDeprecationWarning:

The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.

alternative="'density'", removal="3.1")

Out[15]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x116b0c6a0>

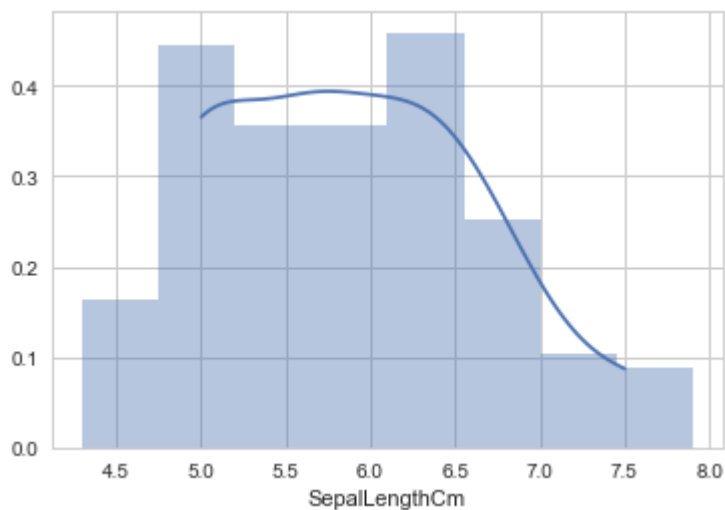




In [16]:

```
1  # En choisissant uniquement une partie de la courbe
2  sns.set(style="whitegrid")
3
4  ax=sns.distplot(df["SepalLengthCm"],
5                  kde=True, kde_kws={'clip': (5, 7.5)})
6
7
8  plt.show()
```

/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/matplotlib/axes/\_axes.py:6521: MatplotlibDeprecationWarning:  
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.  
alternative="'density'", removal="3.1")



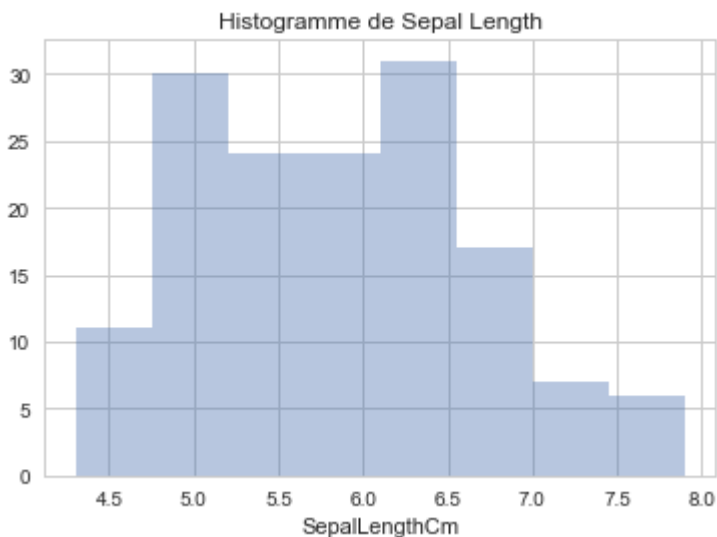
In [17]:

```
1  # Avec un titre
2  # Pour avoir un fond blanc avec grille
3  sns.set(style="whitegrid")
4
5  ax=sns.distplot(df["SepalLengthCm"], kde=False)
6  ax.set_title("Histogramme de Sepal Length")
7
8  plt.show()
```

/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/matplotlib/axes/\_axes.py:6521: MatplotlibDeprecationWarning:

The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.

alternative="'density'", removal="3.1")



In [18]:

```

1 f, axes = plt.subplots(2, 2, figsize=(10, 10), sharex=False)
2 sns.distplot(df["SepalLengthCm"], ax=axes[0, 0])
3 sns.distplot(df["SepalWidthCm"], ax=axes[0, 1])
4 sns.distplot(df["PetalLengthCm"], ax=axes[1, 0])
5 sns.distplot(df["PetalWidthCm"], ax=axes[1, 1])

```

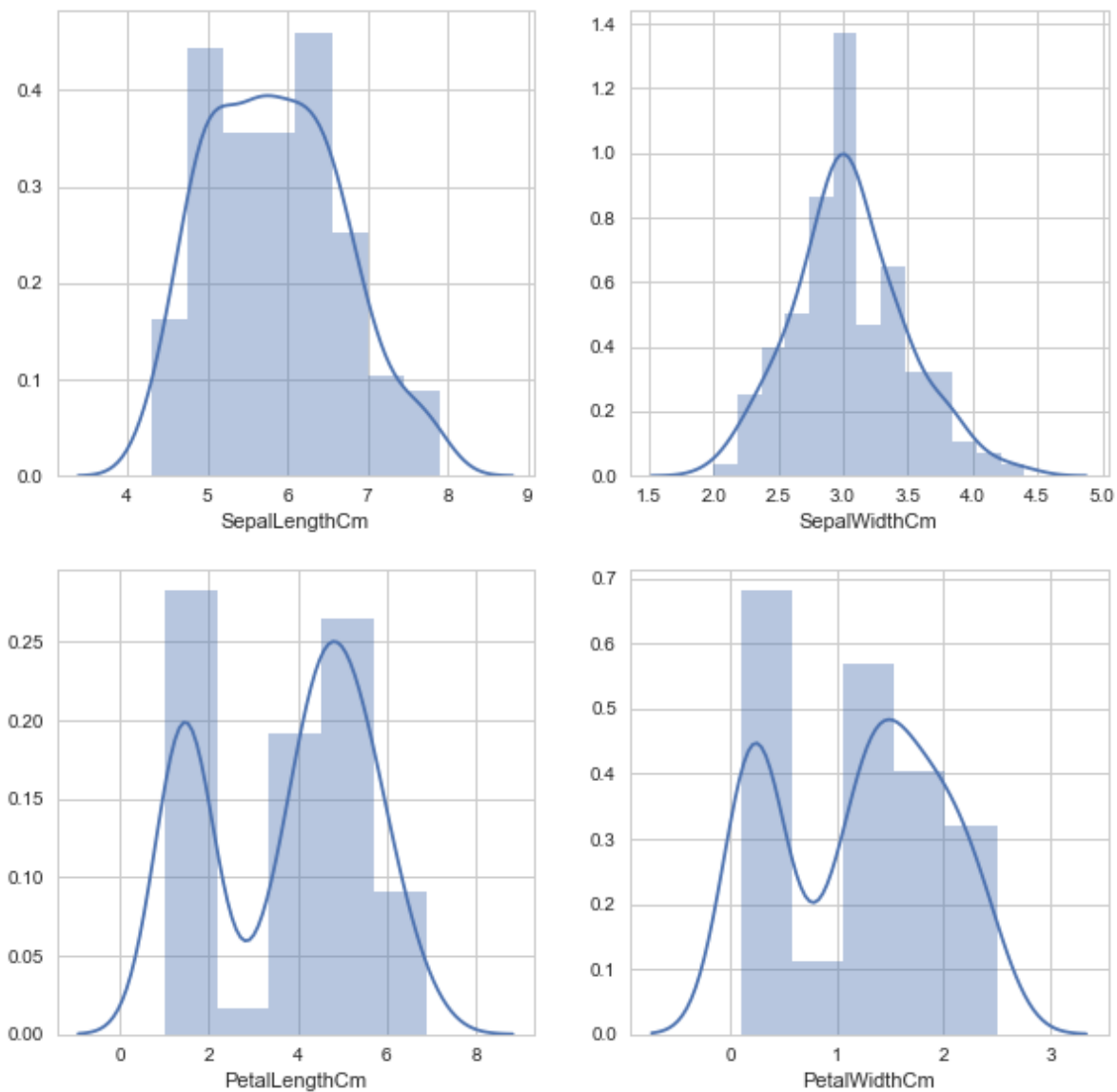
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/site-packages/matplotlib/axes/\_axes.py:6521: MatplotlibDeprecationWarning:

The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' instead.

alternative="'density'", removal="3.1")

Out[18]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1152ae1d0>

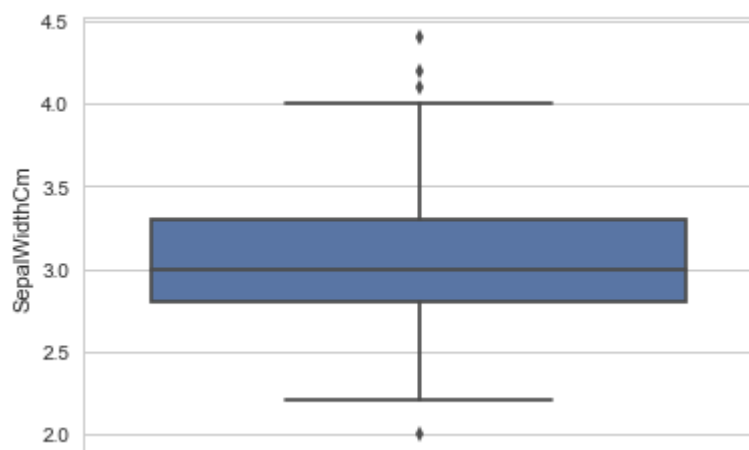


### Boîtes à moustache

Il existe différentes manières de représenter les boîtes à moustache en seaborn.

In [19]:

```
1  ▼ #sns.set(style="whitegrid")  
2    # orient pour mettre la vue en vertical  
3    ax = sns.boxplot(x=df["SepalWidthCm"], orient='v')  
4
```

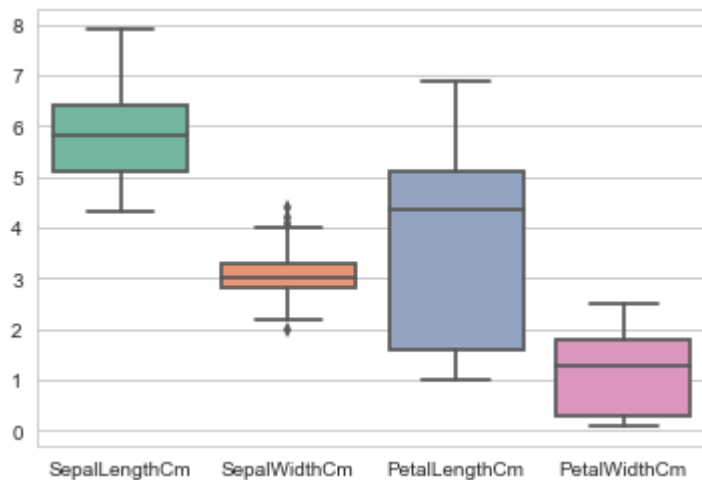


In [20]:

```

1  ▼ # palette = 'Set2' pour changer les couleurs
2    ax = sns.boxplot(data=df, palette='Set2')

```



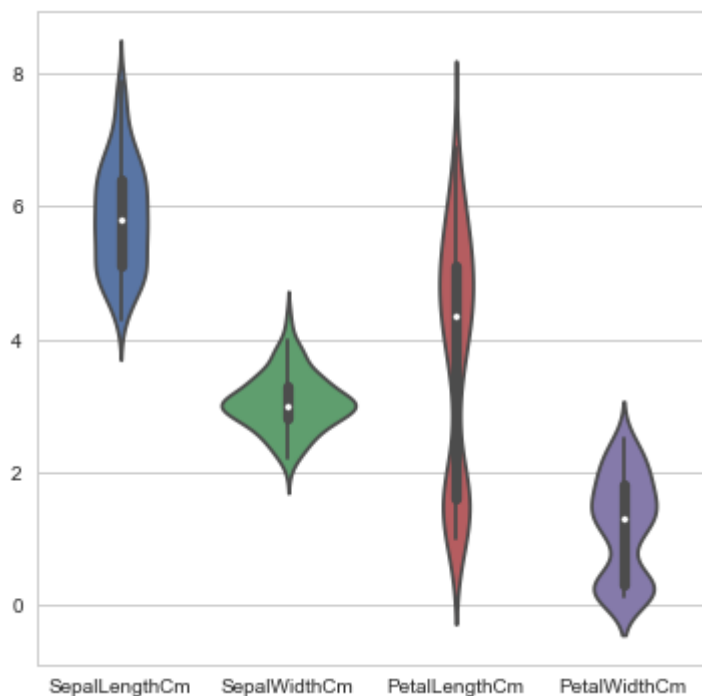
Le mode violon est une combinaison de boîte à moustache et d'estimation de la densité.

In [21]:

```

1  ▼ # changer la taille de la figure
2    fig, ax = plt.subplots(figsize=(6,6))
3    # Affichage en violon
4    ax=sns.violinplot(data=df)
5

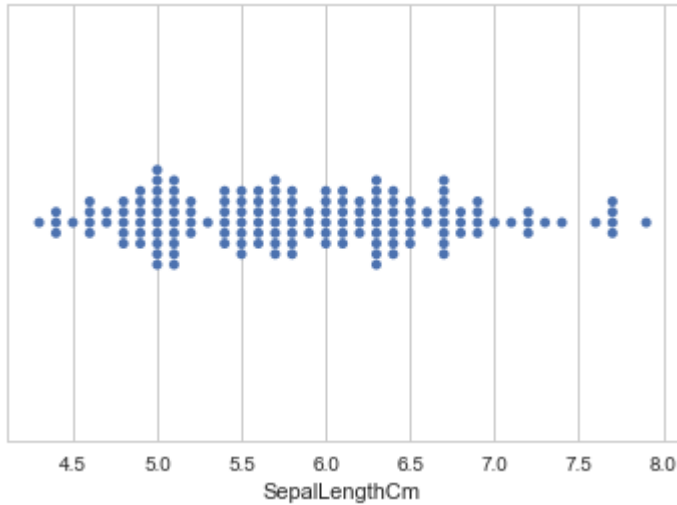
```



Les swarmplot indiquent une bonne représentation de la distribution des données (elles sont toutes présentées sans chevauchement) mais sont difficile à interpréter lorsqu'il y a beaucoup de données.

In [22]:

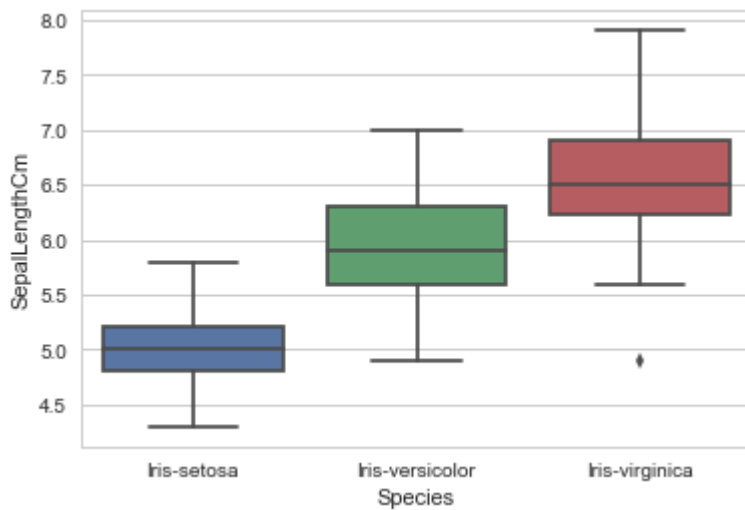
```
1 ax = sns.swarmplot(x=df["SepalLengthCm"])
```



Il est possible de connaître les boîtes à mouchoir par rapport à une variable catégorielle.

In [23]:

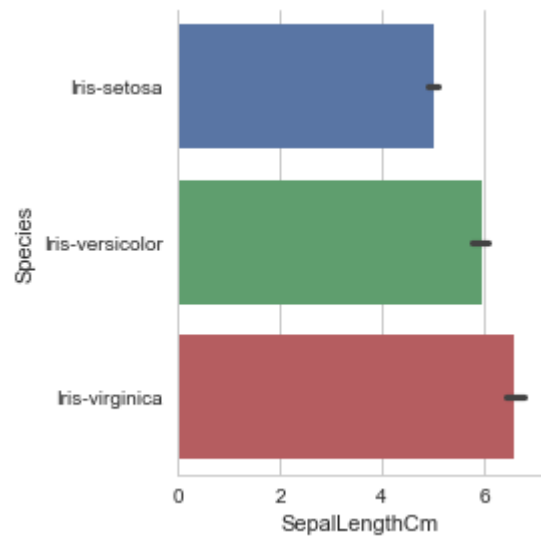
```
1 ax = sns.boxplot(x="Species", y="SepalLengthCm", data=df)
```



## Travailler avec plusieurs attributs

In [24]:

```
1  sns.factorplot("SepalLengthCm", 'Species',
2                    data=df, kind="bar",  legend=False)
3
4  # Show plot
5  plt.show()
```



In [25]:

```
1  corr = df.corr()
2  corr
```

Out[25]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [26]:

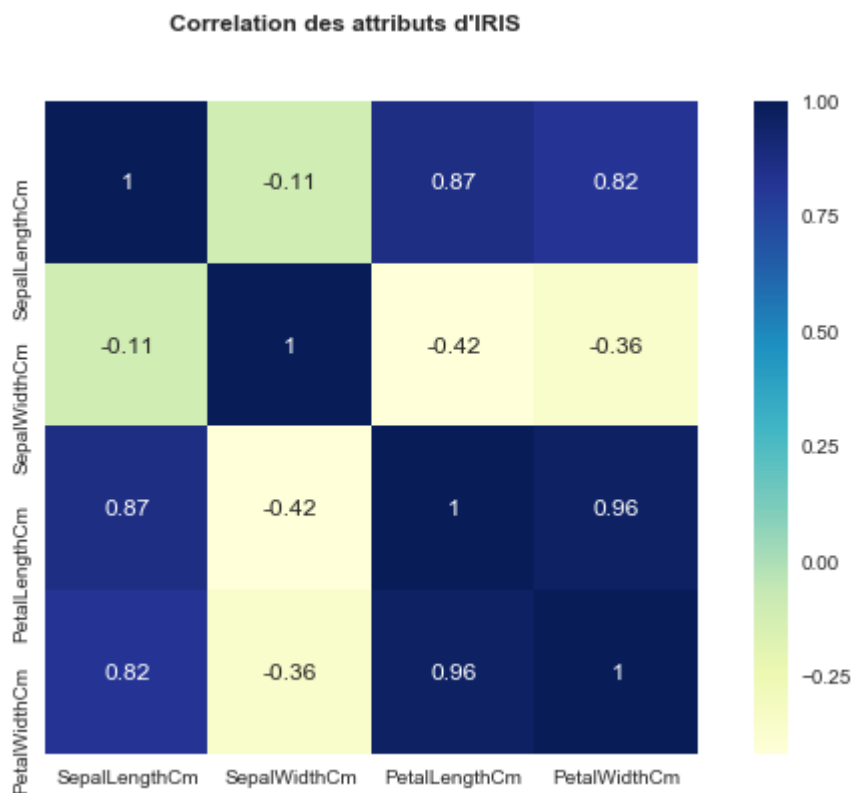
```

1  fig, ax = plt.subplots(1, 1, figsize=(10,6))
2
3  ▼ hm = sns.heatmap(corr,
4                      ax=ax,          # Axes où afficher
5                      cmap="YlGnBu", # Couleur
6                      square=True,   # Si True, toutes les cellules ont le même a
7                      annot=True     # Pour afficher les valeurs
8                      )
9
10 ▼ fig.suptitle("Correlation des attributs d'IRIS",
11               fontsize=12,
12               fontweight='bold')

```

Out[26]:

```
Text(0.5, 0.98, "Correlation des attributs d'IRIS")
```



## Scatter plot matrix

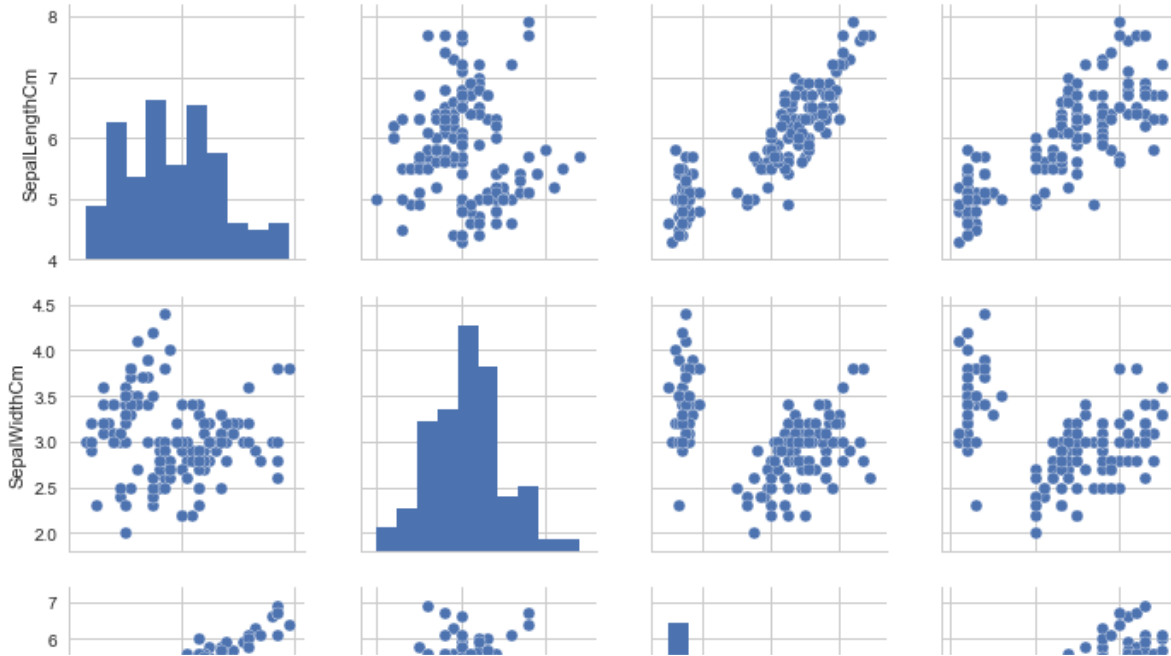


In [27]:

```

1
2
3 # Matrice de base
4 g = sns.pairplot(df)
5
6

```



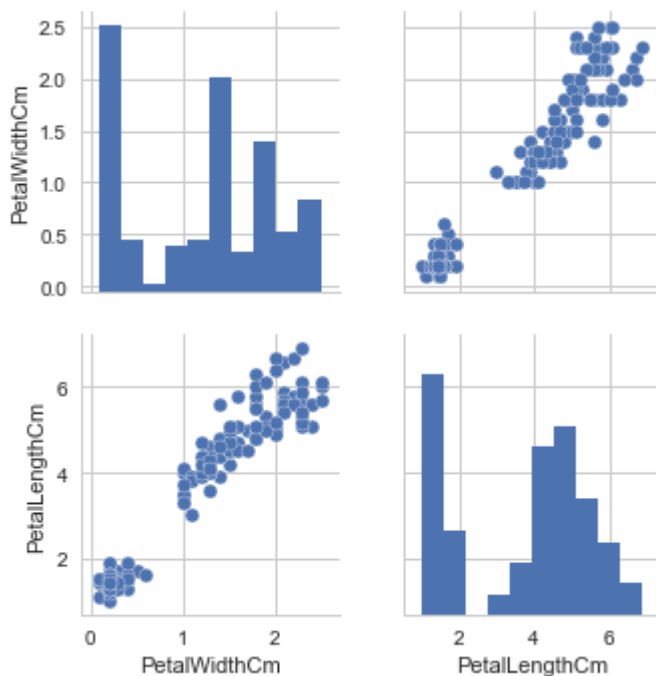
In [28]:

```

1 ▼ # Matrice réduite a un sous ensemble d'attributs
2 print ("Un exemple de variables fortement corrélées")
3 ▼ g = sns.pairplot(df, vars=["PetalWidthCm",
4                             "PetalLengthCm"])
5
6

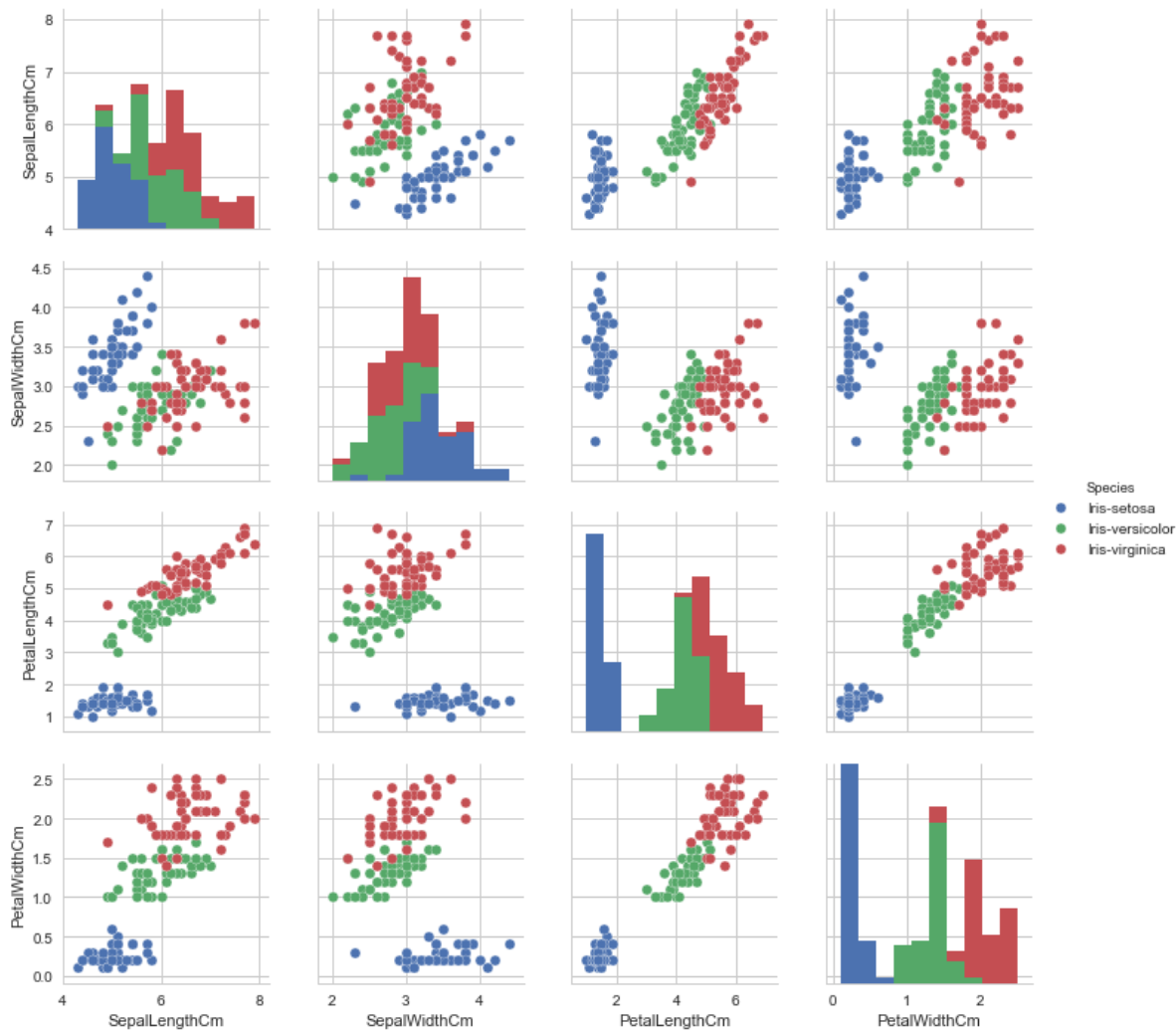
```

Un exemple de variables fortement corrélées



In [29]:

```
1  ▾ # En mettant en avant les espèces
2  g = sns.pairplot(df,hue="Species")
```



In [30]:

```
1  # En mettant en avant une estimation de la densité
2  g = sns.pairplot(df, hue="Species", diag_kind="kde")
```

