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 maplotlib

In [1]:

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

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

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, **kwds)

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]:

```
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris
names = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',

df = pd.read_csv(url, names=names)
# 5 premières lignes du fichier
df.head()

```

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

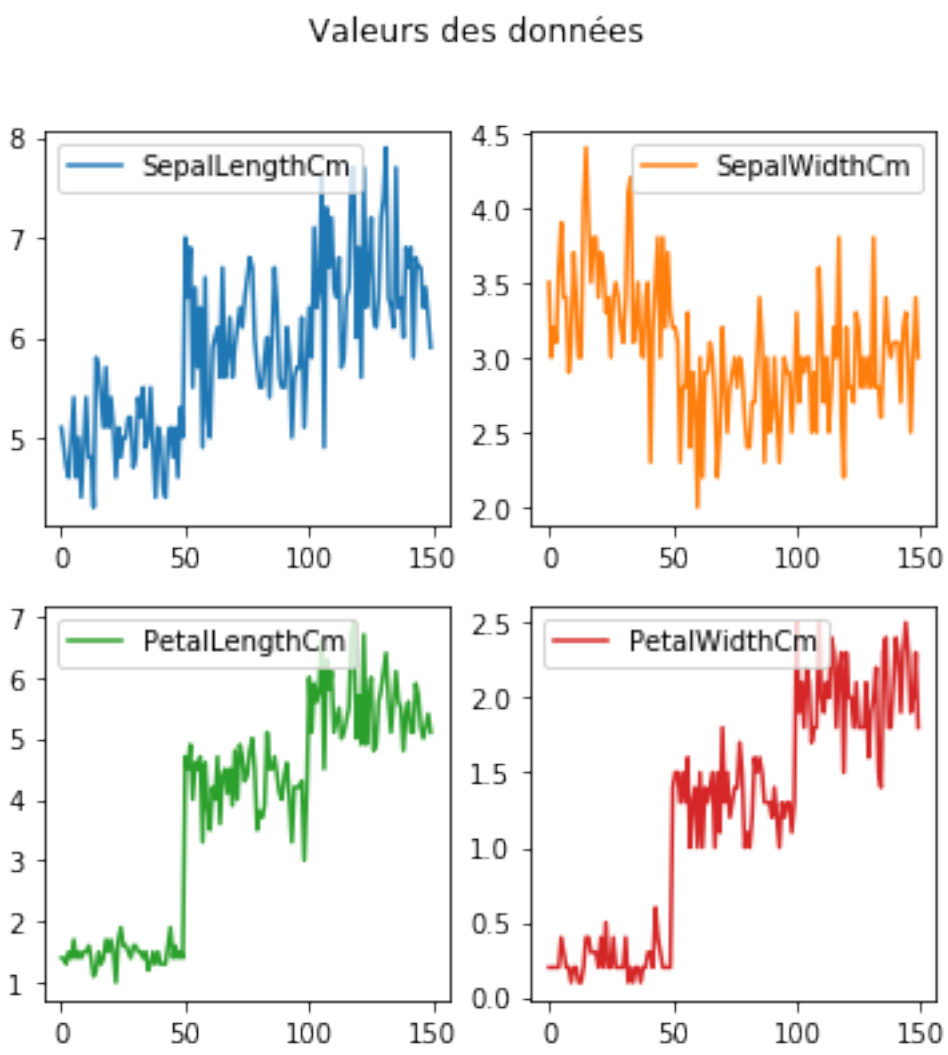
Dans cette partie nous considérons chaque attribut indépendamment.

Affichage des valeurs

Pour avoir une première idée des données manipulées

In [3]:

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

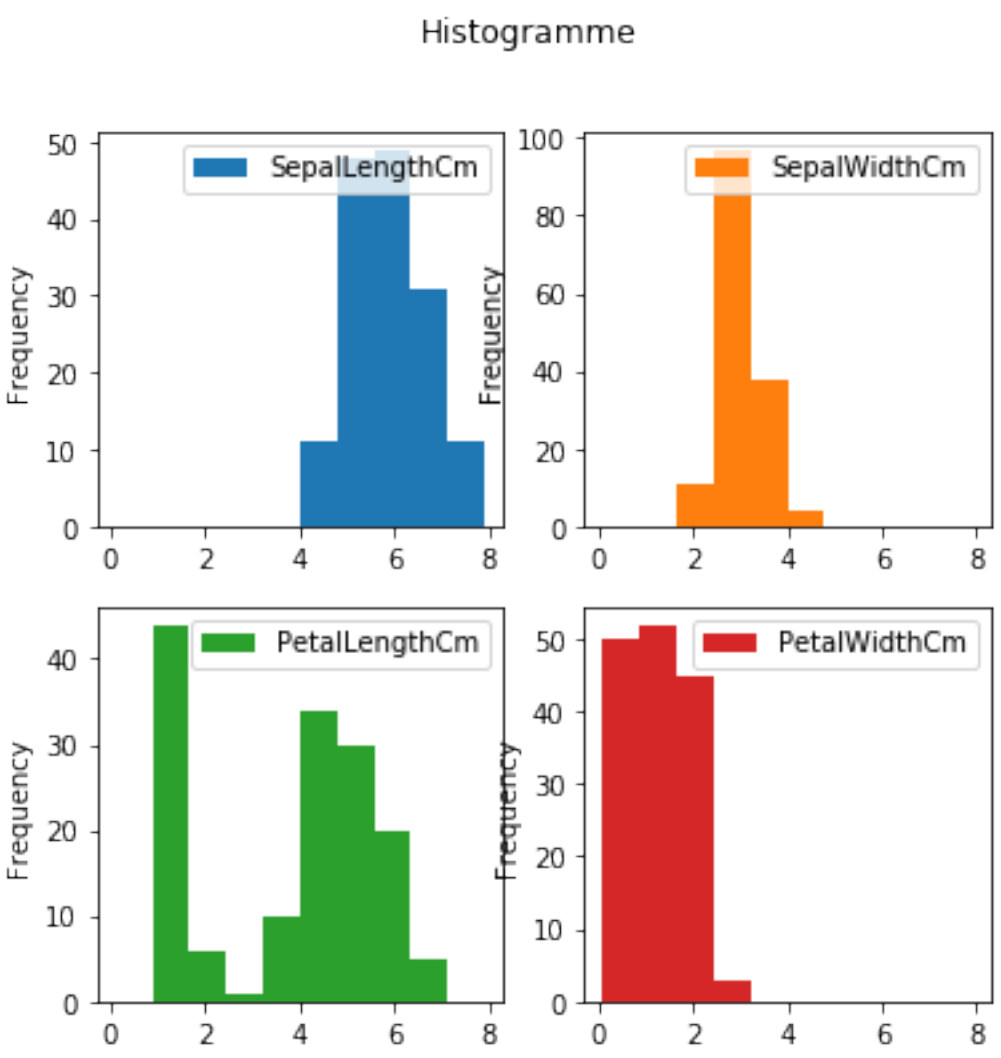


Histogramme

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

In [4]:

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

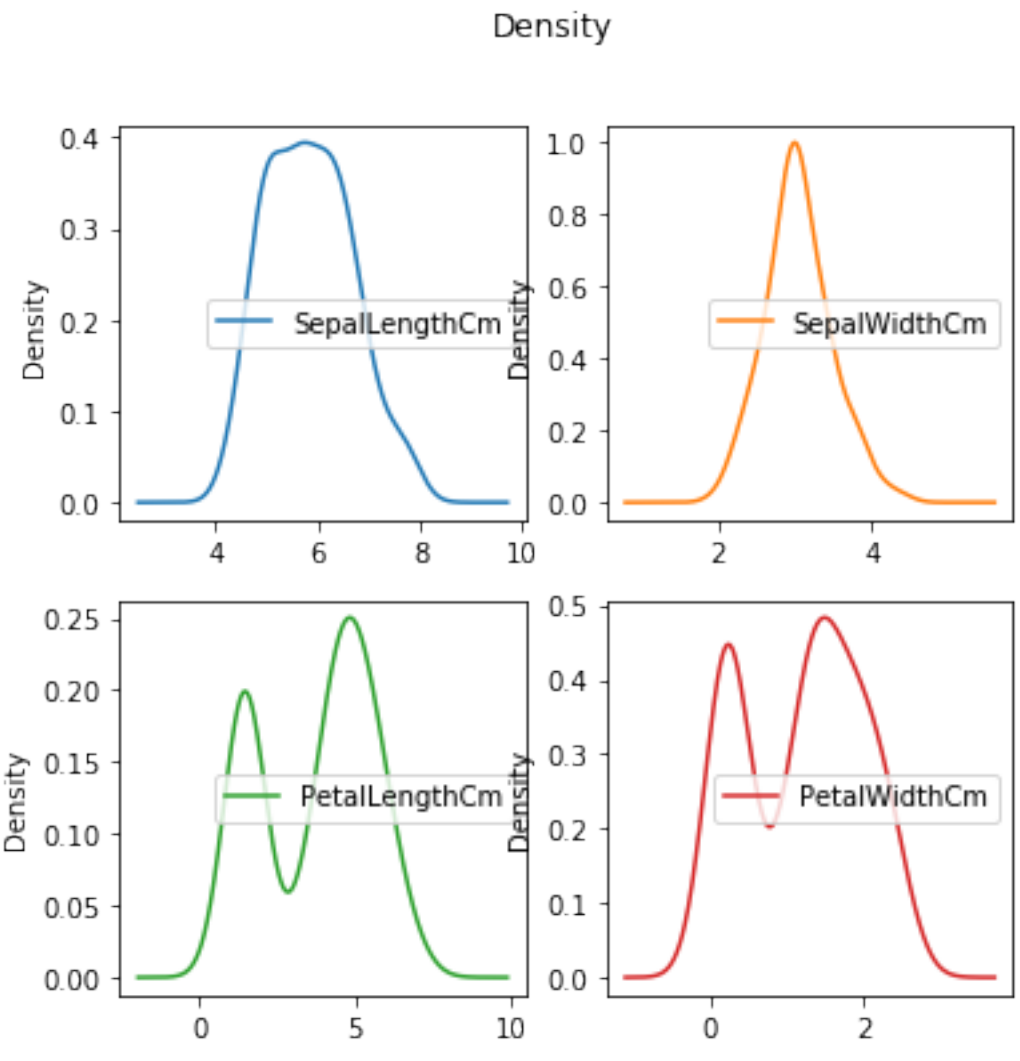


Affichage des densités

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

In [5]:

```
df.plot(kind='density', subplots=True, layout=(2,2), figsize=(6,6),sharex=I
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]:

```
df.plot(kind='box', subplots=True, layout=(2,2), figsize=(6,6),sharex=False
plt.show()
```

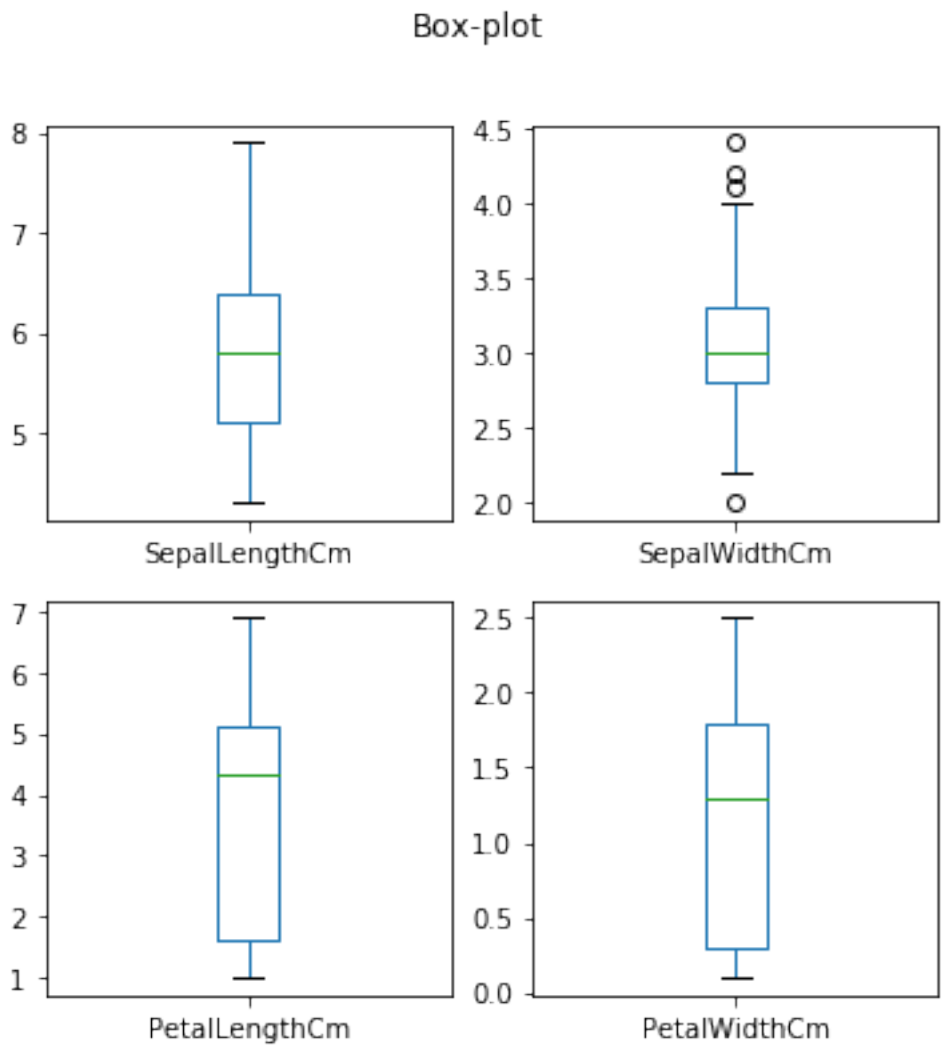
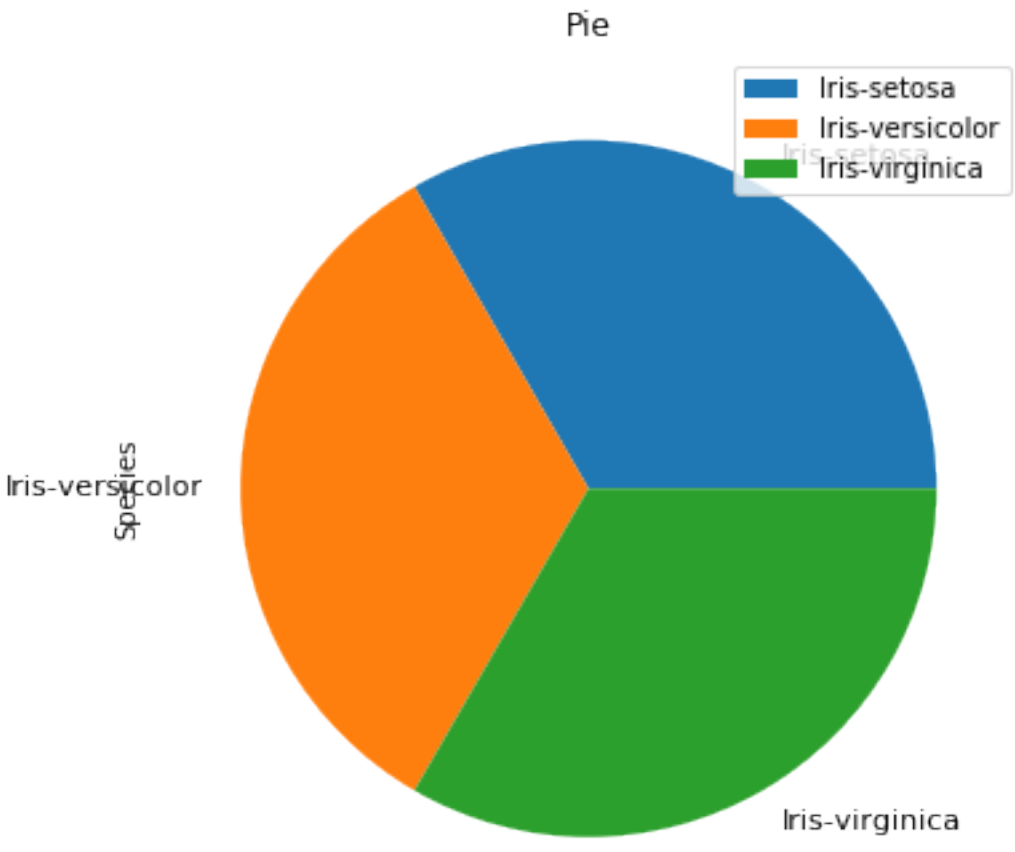


Diagramme circulaire

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

In [7]:

```
# Species n'est pas une valeur numérique, ici on compte combien de valeurs
# fontsize pour changer la taille de la font
# legend=True pour afficher la légende
df['Species'].value_counts().plot(kind='pie', figsize=(6,6),title='Pie', fo
plt.show()
```



Travailler avec plusieurs attributs

Correlations et carte de chaleur

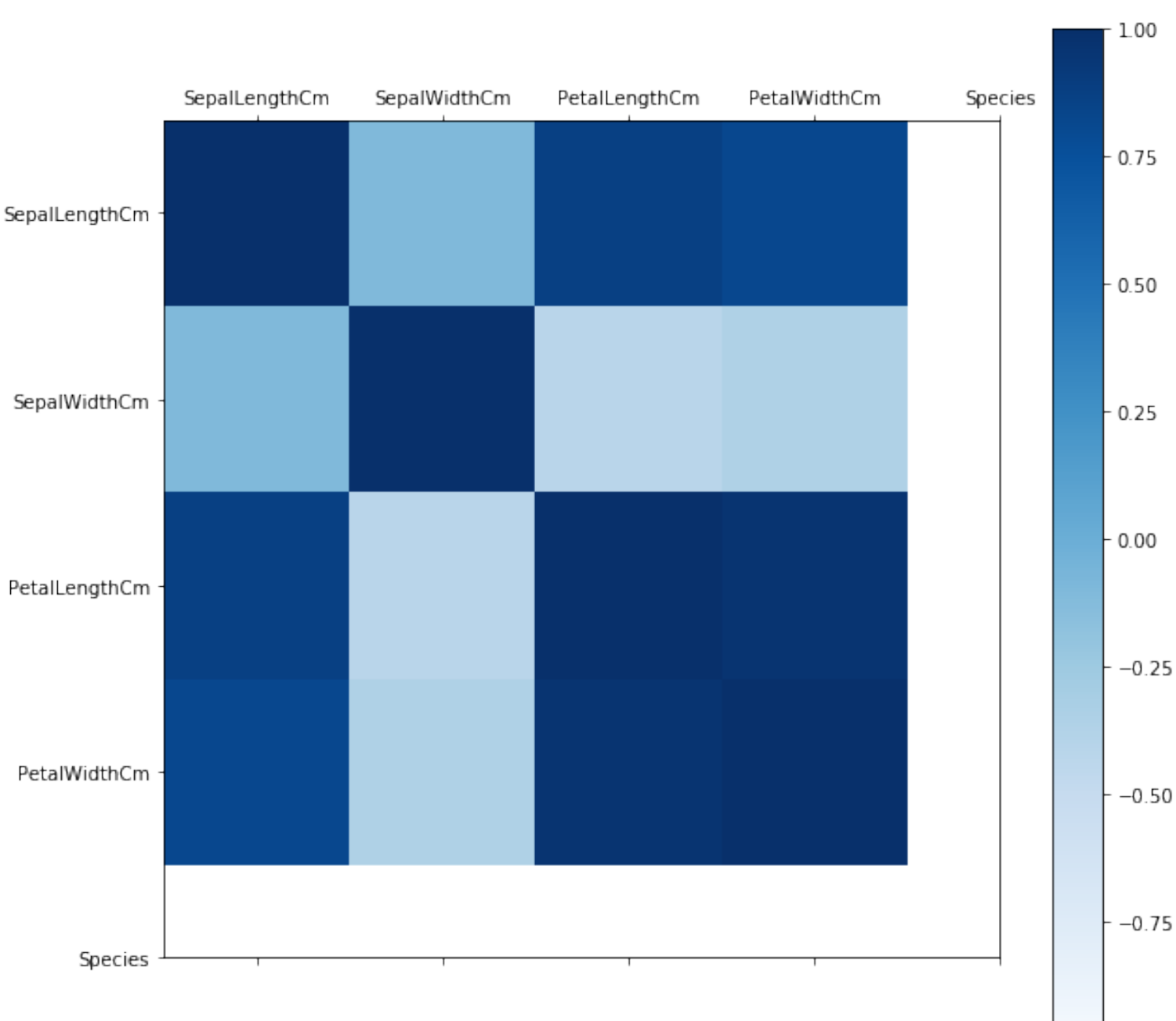
Il est possible via la fonction correlation de déterminer les corrélations qui existent entre les différents attributs. Les cartes de chaleur (heatmap) peuvent être utilisées pour mettre en évidence ces corrélations.

In [8]:

```
import numpy


correlations = df.corr()
# Affichage des valeurs de corrélation
print (correlations)

# Transformation des données de Species en données catégorielles
# pour pouvoir calculer aussi les corrélations avec les autres
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y=df['Species']
le.fit(y)
df['Species'] = le.transform(y)


# Affichage de la matrice de corrélation
```

```
# Affichage de la matrice de correlation
fig = plt.figure(figsize=(10,10))
# Partie à droite pour montrer les variations de valeurs
ax = fig.add_subplot(111)
# Détermine les valeurs maximales et minimales à afficher et le thème Blues
cax = ax.matshow(correlations, vmin=-1, vmax=1, cmap=plt.cm.Blues)
# Récupération des couleurs en fonctions des valeurs
fig.colorbar(cax)
# Le tableau principal
# debute à 0, taille 5 (5 valeurs à afficher), largeur d'une colonne 1
ticks = numpy.arange(0,5,1)
# mise en place
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(names)
ax.set_yticklabels(names)
plt.show()
# La diagonale est forcément la plus foncée
```

```
                SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidt
hCm
SepalLengthCm        1.000000     -0.109369       0.871754       0.817
954
SepalWidthCm        -0.109369      1.000000      -0.420516      -0.356
544
PetalLengthCm        0.871754     -0.420516       1.000000       0.962
757
PetalWidthCm         0.817954     -0.356544       0.962757       1.000
000
```



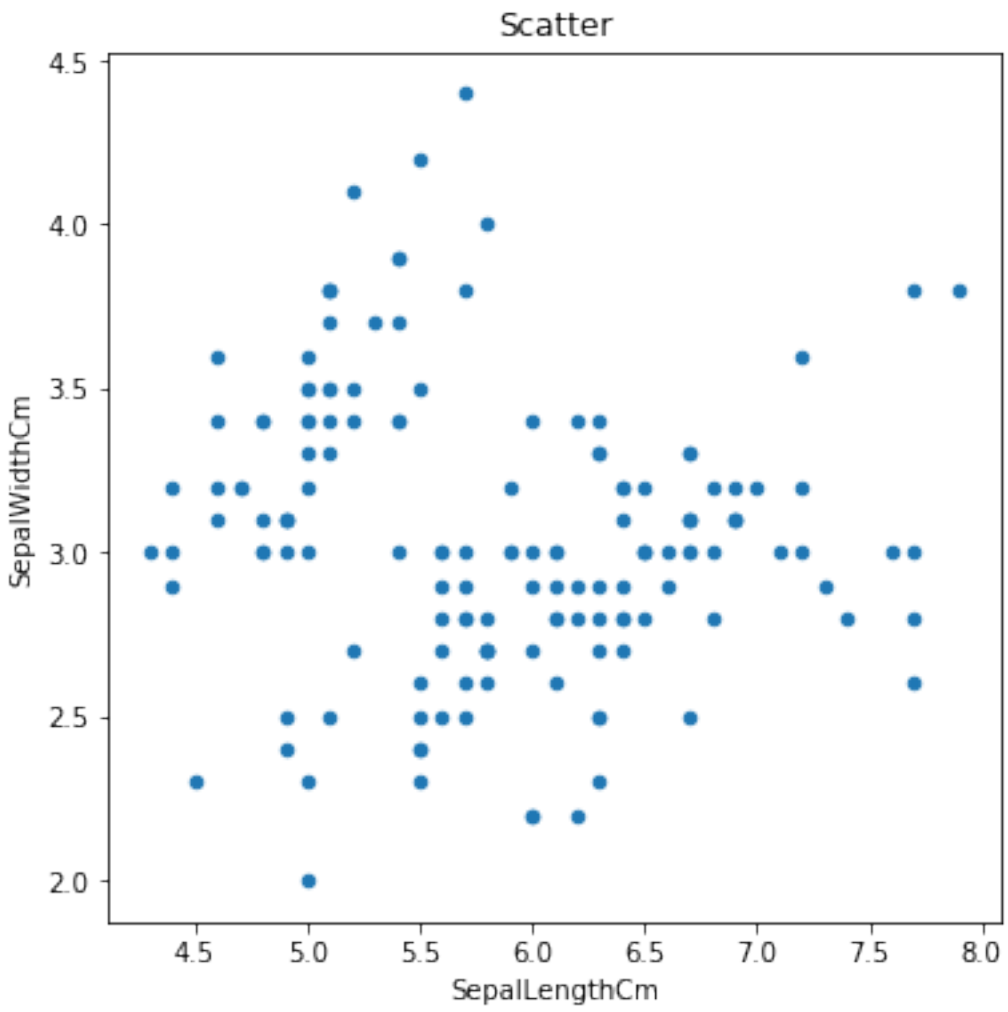
−1.00

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.

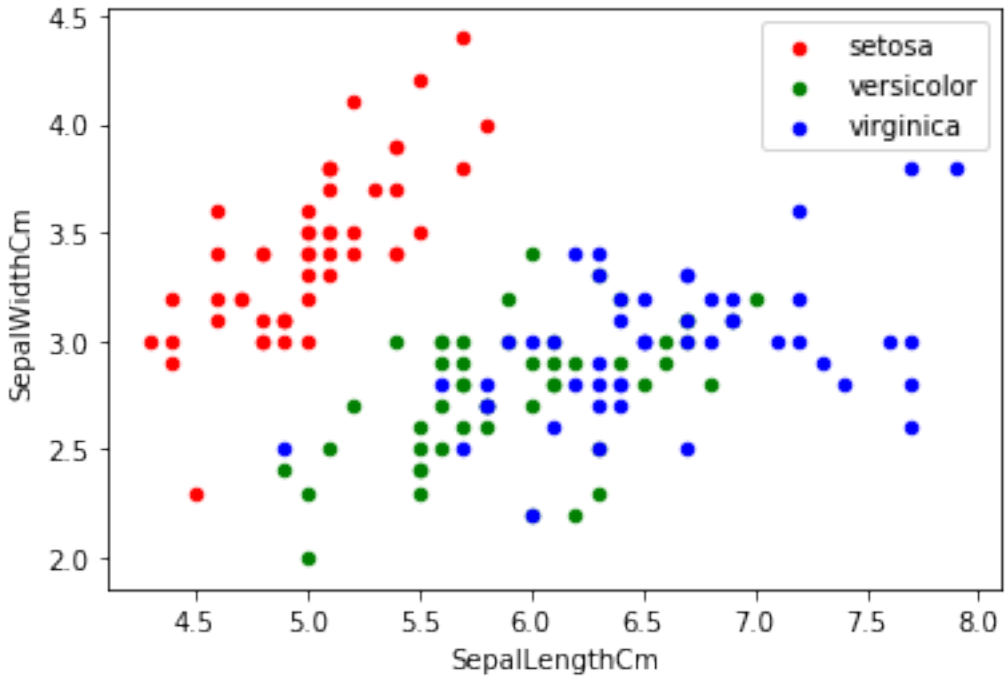
In [9]:

```
#df.plot(kind='scatter', subplots=True, layout=(2,2), figsize=(6,6),sharex=
# Pour deux attributs il faut préciser le x et le y
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris
names = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',

df = pd.read_csv(url, names=names)
df.plot(kind='scatter', x='SepalLengthCm', y='SepalWidthCm',figsize=(6,6),s
plt.show()

# Pour mettre en avant les différentes espèces
ax = df[df.Species=='Iris-setosa'].plot(kind='scatter', x='SepalLengthCm',
                                                    color='red', label='set
df[df.Species=='Iris-versicolor'].plot(kind='scatter', x='SepalLengthCm', y
                                                  color='green', label='versi
df[df.Species=='Iris-virginica'].plot(kind='scatter', x='SepalLengthCm', y=
                                                  color='blue', label='virgi
ax.set_title("Scatter")

```



Out[9]:

```
Text(0.5, 1.0, 'Scatter')
```



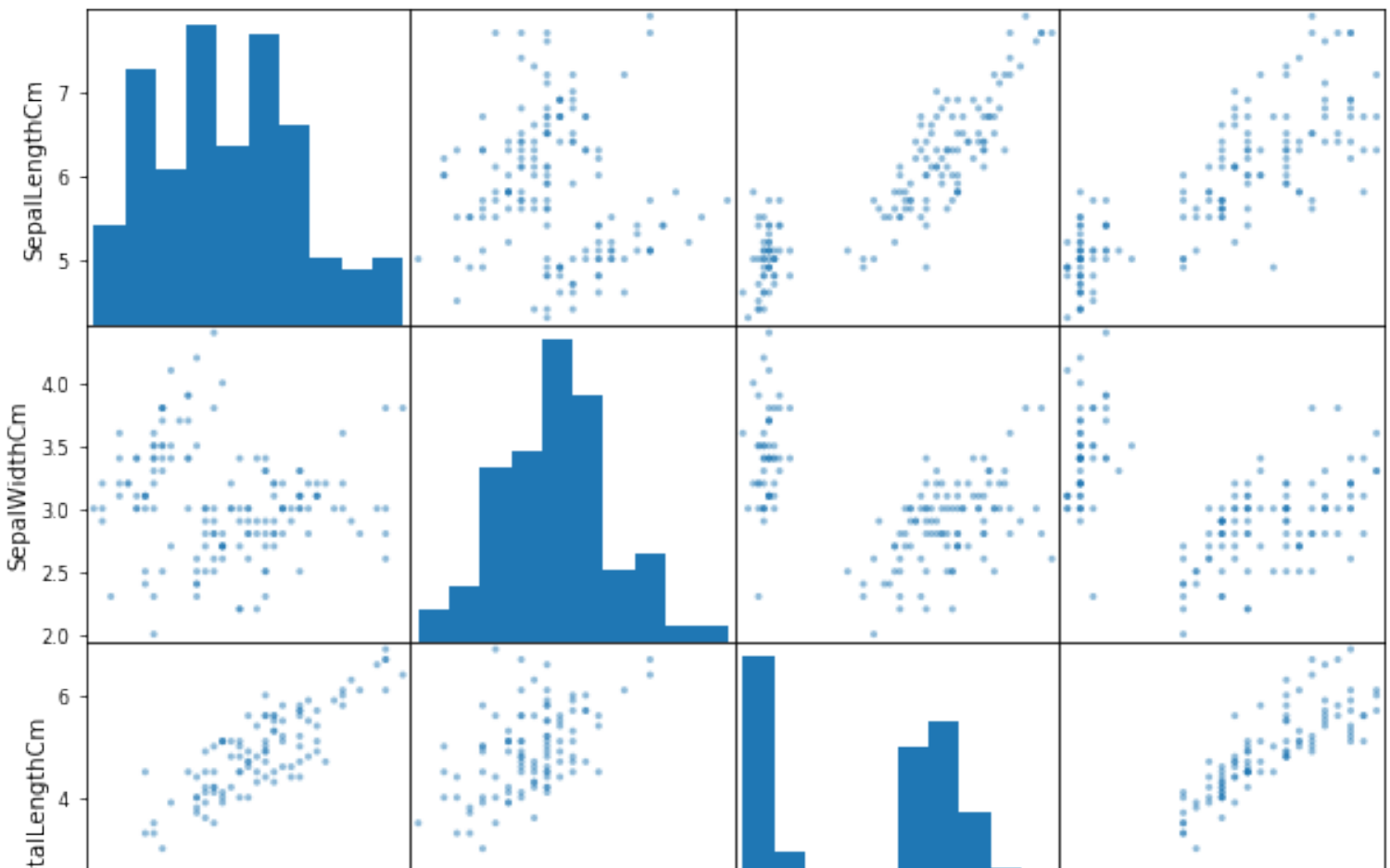
Scatter plot matrix

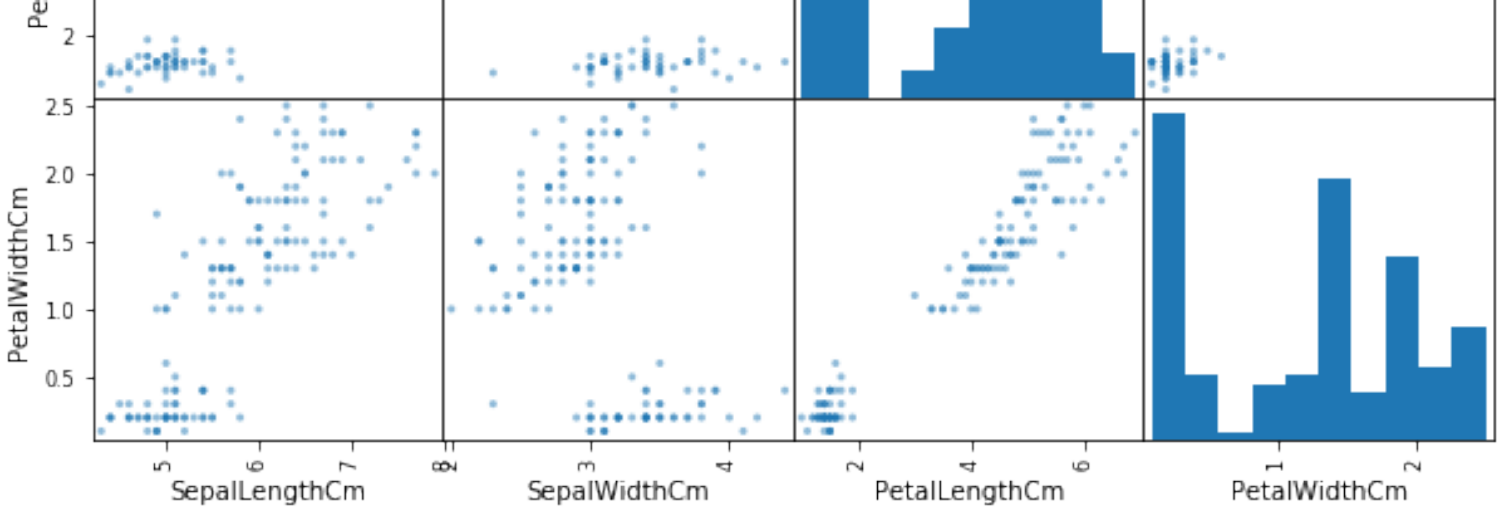
Cette matrice affiche toutes les valeurs en croisant les variables deux à deux.

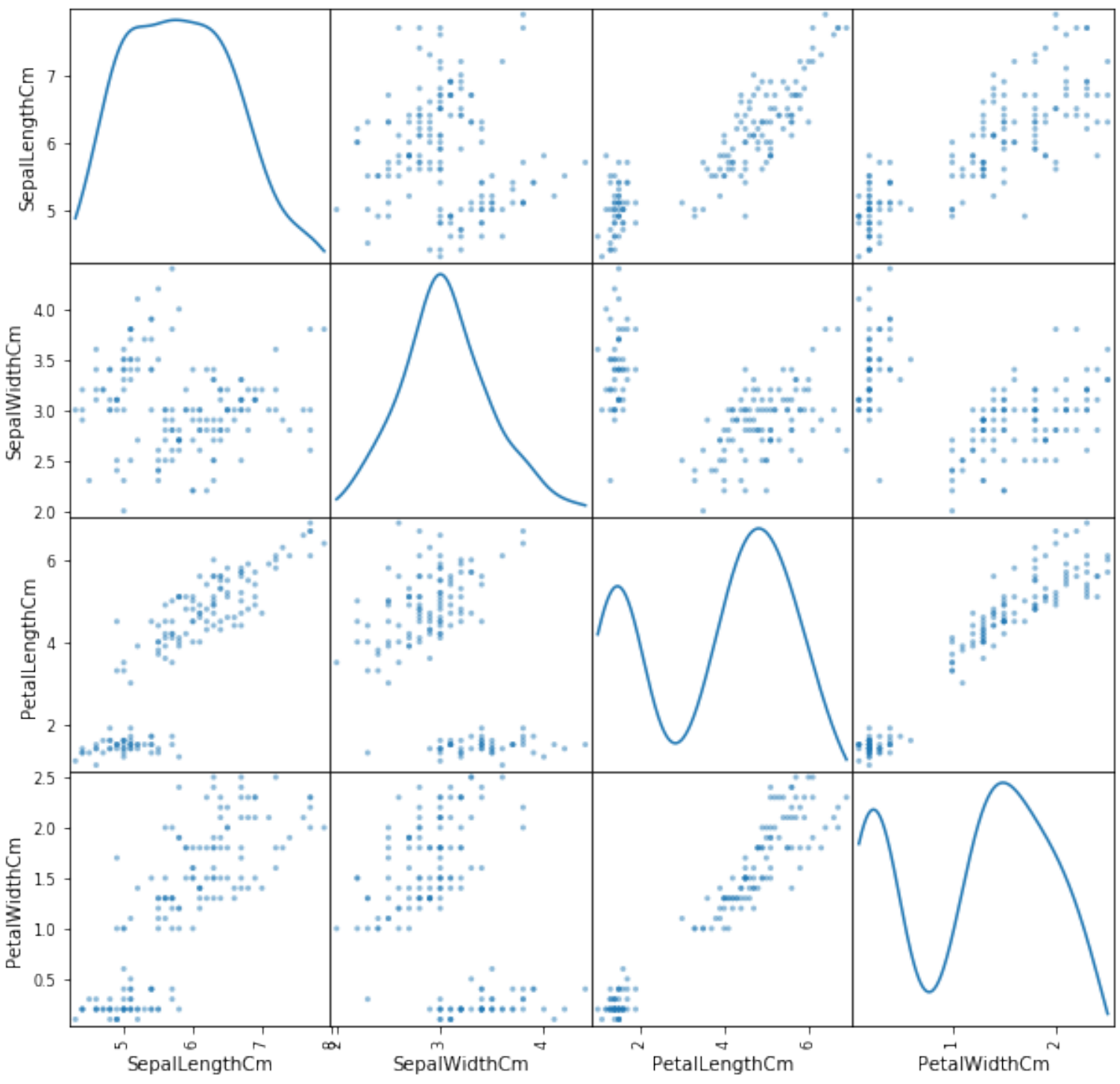
Dans pandas il est possible d'utiliser la fonction scatter_matrix (voir [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.plotting.scatter_matrix.html (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.plotting.scatter_matrix.html)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.plotting.scatter_matrix.html))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

In [10]:

```
from pandas.plotting import scatter_matrix
# Pour afficher toute la matrice
scatter_matrix(df,figsize=(10,10))
plt.show()

# Pour afficher la matrice mais avec les densités plutôt que des histogram
scatter_matrix(df,figsize=(10,10), diagonal='kde')
plt.show()
```







Utilisation de Seaborn

Seaborn est une librairie plus récente qui permet de pouvoir améliorer les visualisations. Une documentation complète est disponible : [https://seaborn.pydata.org (https://seaborn.pydata.org)](https://seaborn.pydata.org).
Il faut importer la librairie seaborn

In [11]:

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

In [12]:

```
import pandas as pd
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris
names = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',

df = pd.read_csv(url, names=names)
# 5 premières lignes du fichier
df.head()


```

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

Histogramme

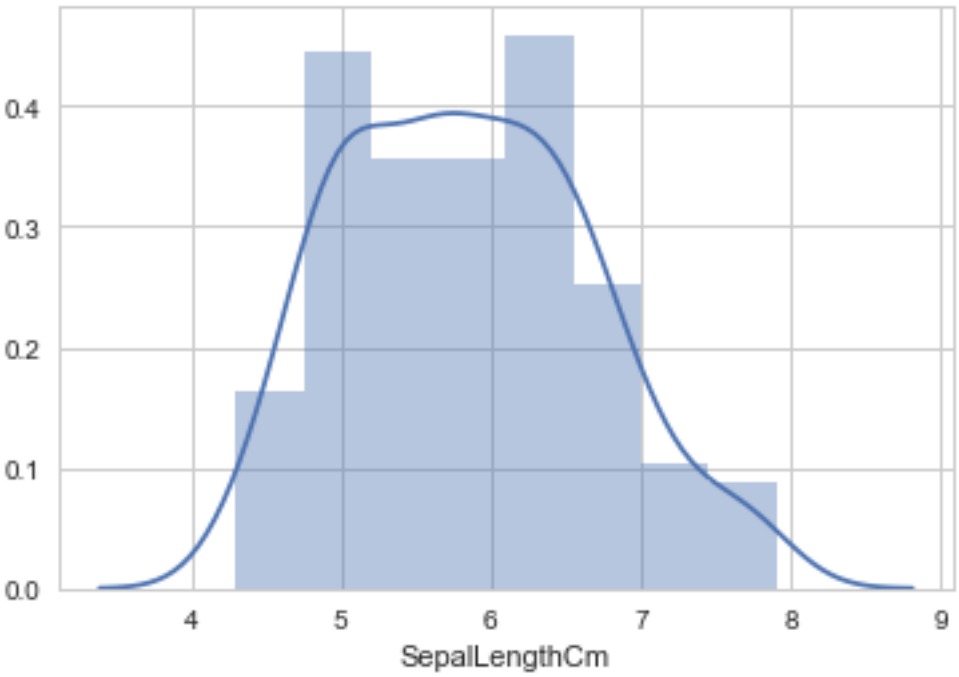
In [13]:

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

```

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

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x117ed3d68>
```



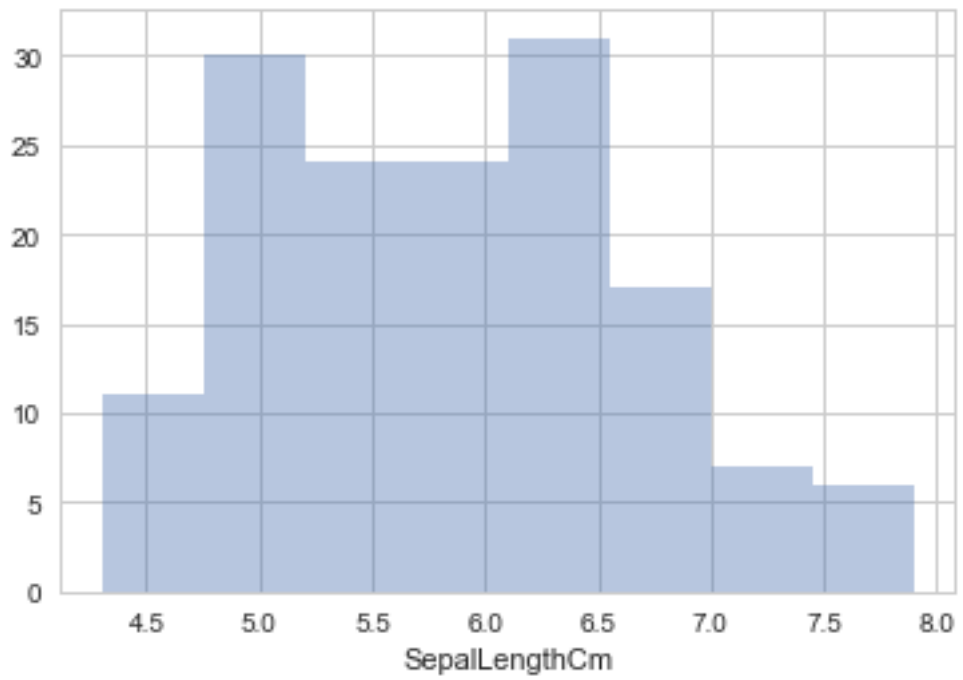
In [14]:

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

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

Out[14]:

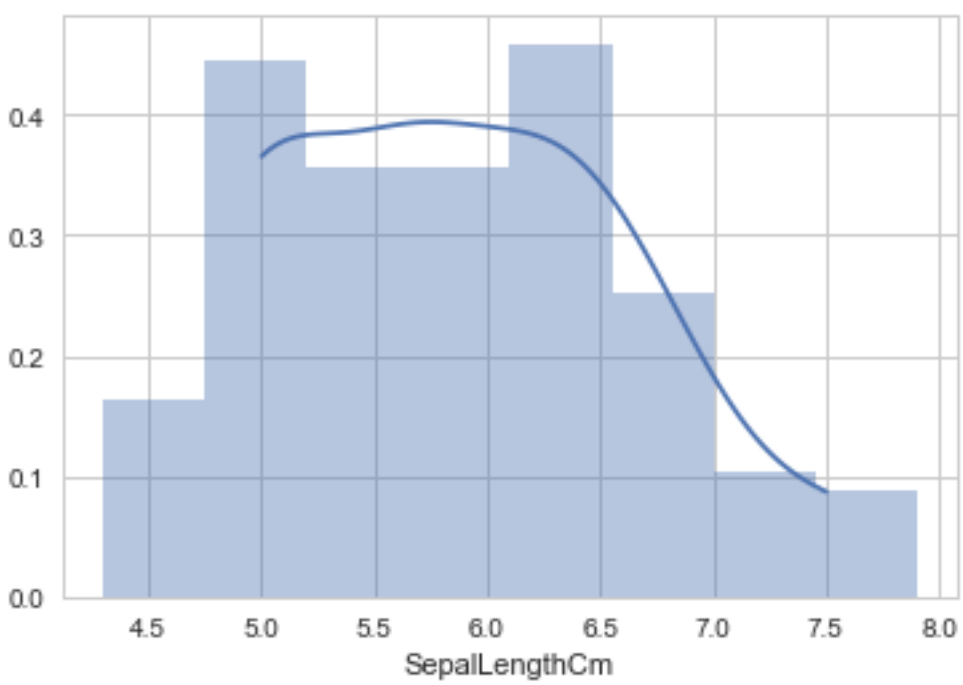
```
<matplotlib.axes._subplots.AxesSubplot at 0x11851b358>
```



In [15]:

```
# En choissant uniquement une partie de la courbe
sns.set(style="whitegrid")

ax=sns.distplot(df["SepalLengthCm"], kde=True, kde_kws={'clip': (5, 7.5)})


plt.show()
```

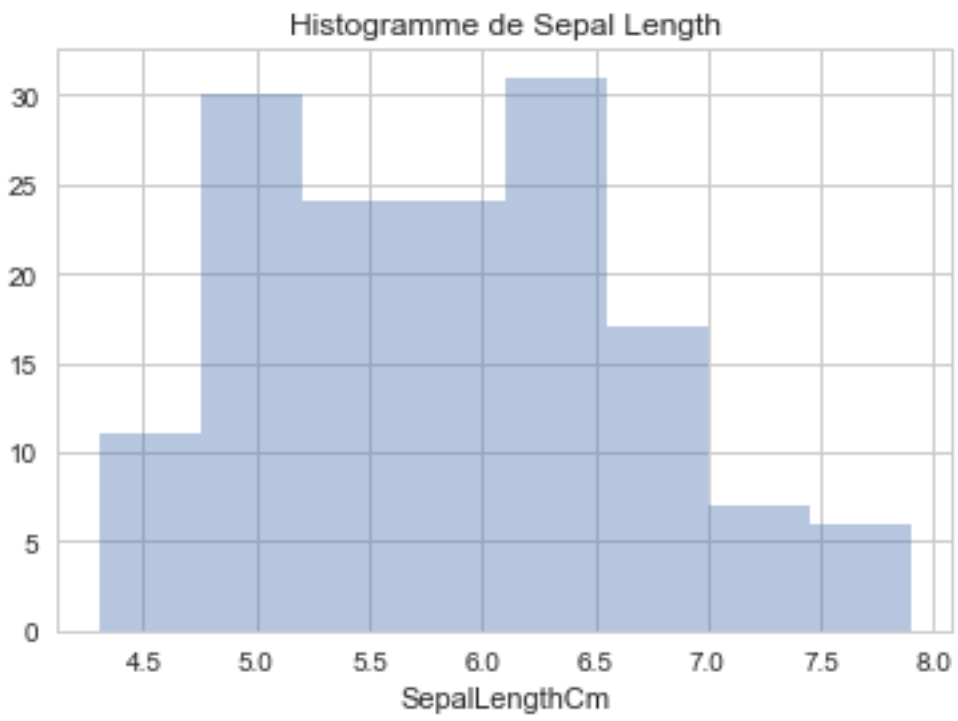
```
/Users/pascalponcelet/Desktop/Sicki-learn/Tools/tools/lib/python3.6/
site-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWa
rning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remo
ved in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
```



In [16]:

```
# Avec un titre
# Pour avoir un fond blanc avec grille
sns.set(style="whitegrid")

ax=sns.distplot(df["SepalLengthCm"], kde=False)
ax.set_title("Histogramme de Sepal Length")

plt.show()
```

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



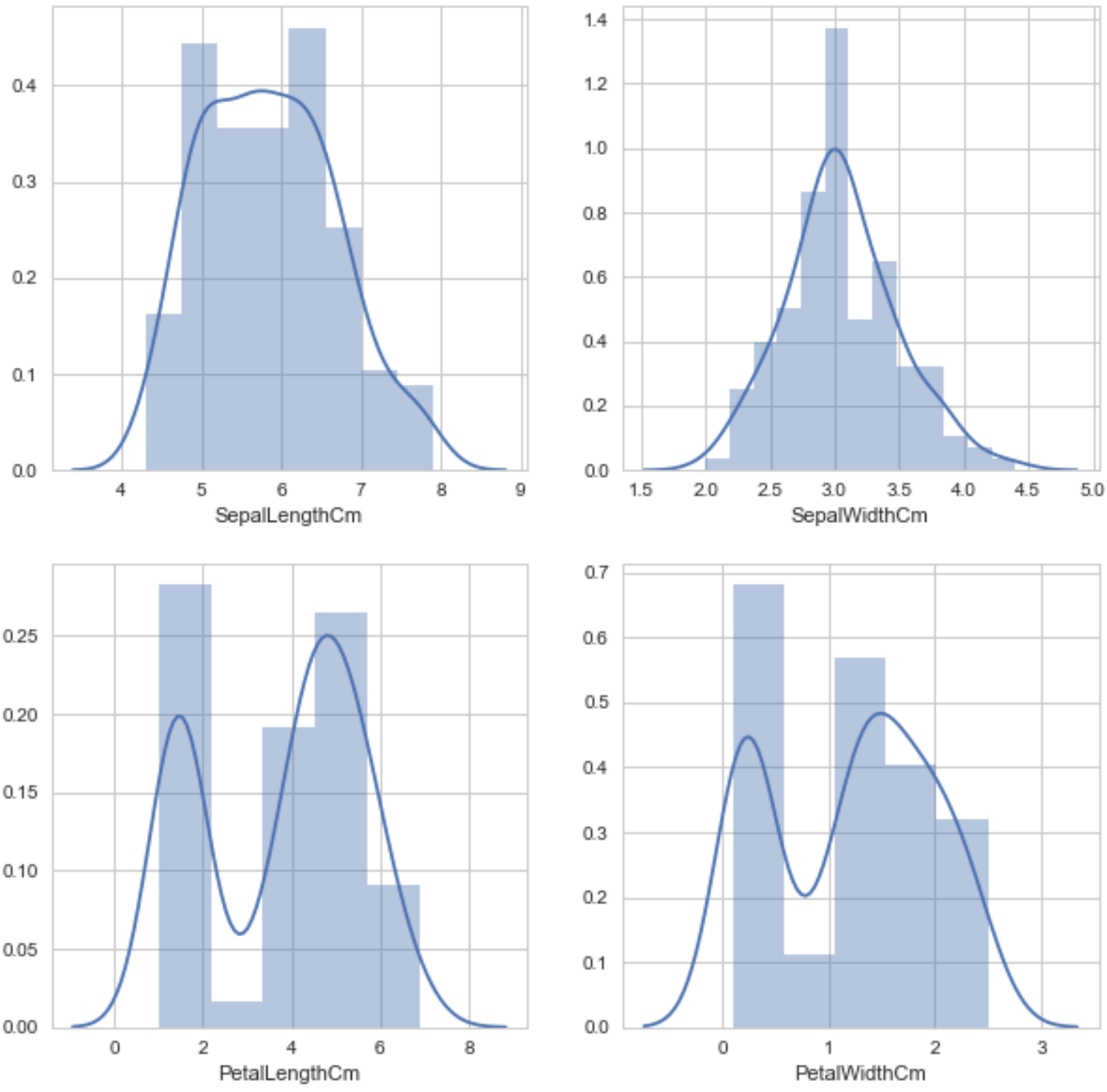
In [17]:

```
f, axes = plt.subplots(2, 2, figsize=(10, 10), sharex=False)
sns.distplot(df["SepalLengthCm"], ax=axes[0, 0])
sns.distplot(df["SepalWidthCm"], ax=axes[0, 1])
sns.distplot(df["PetalLengthCm"], ax=axes[1, 0])
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: MatplotlibDeprecationWa
rning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remo
ved in 3.1. Use 'density' instead.
  alternative="'density'", removal="3.1")
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x115a79e80>
```

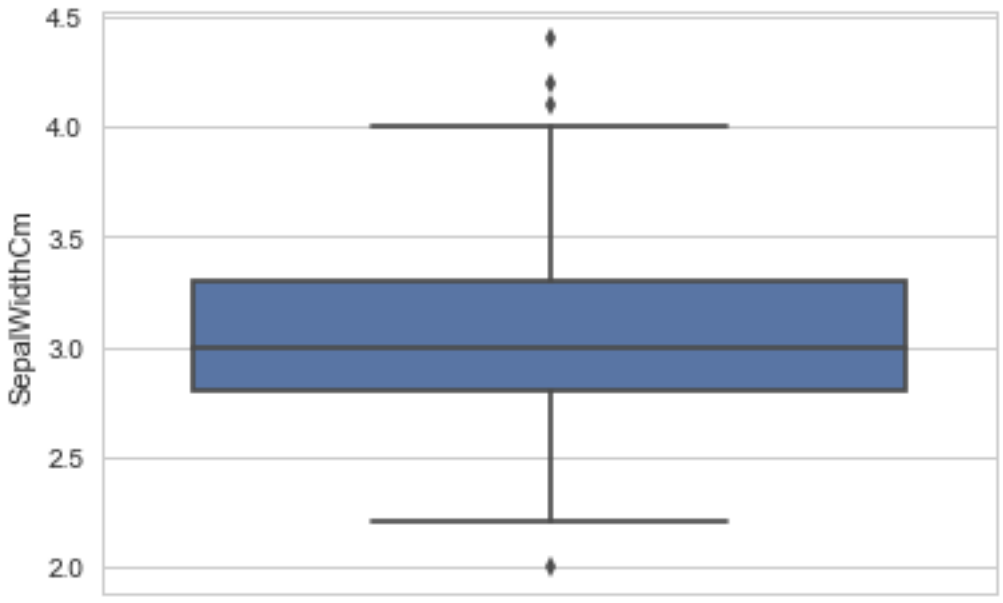


Boîtes à moustache

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

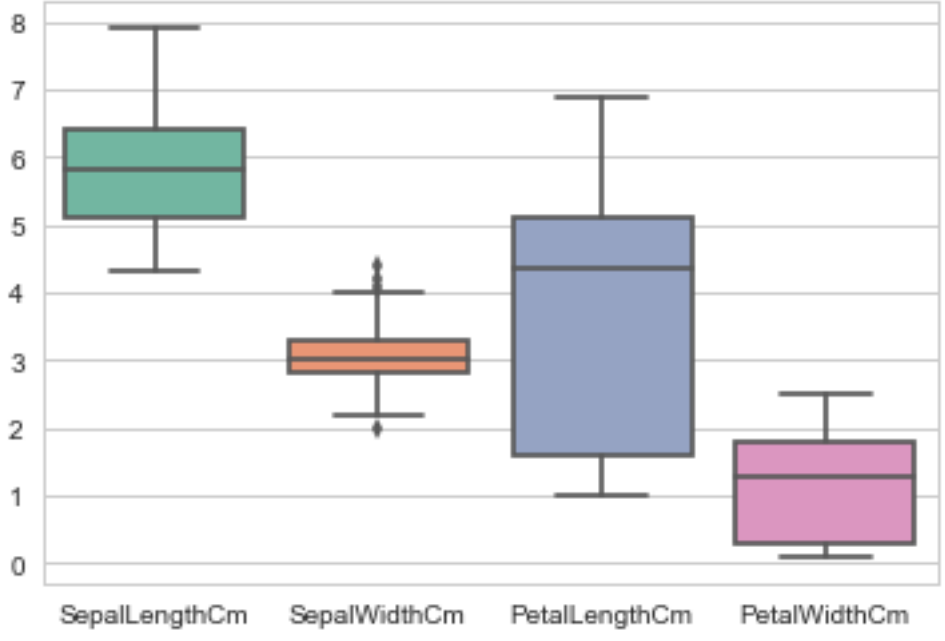
In [18]:

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

```



In [19]:

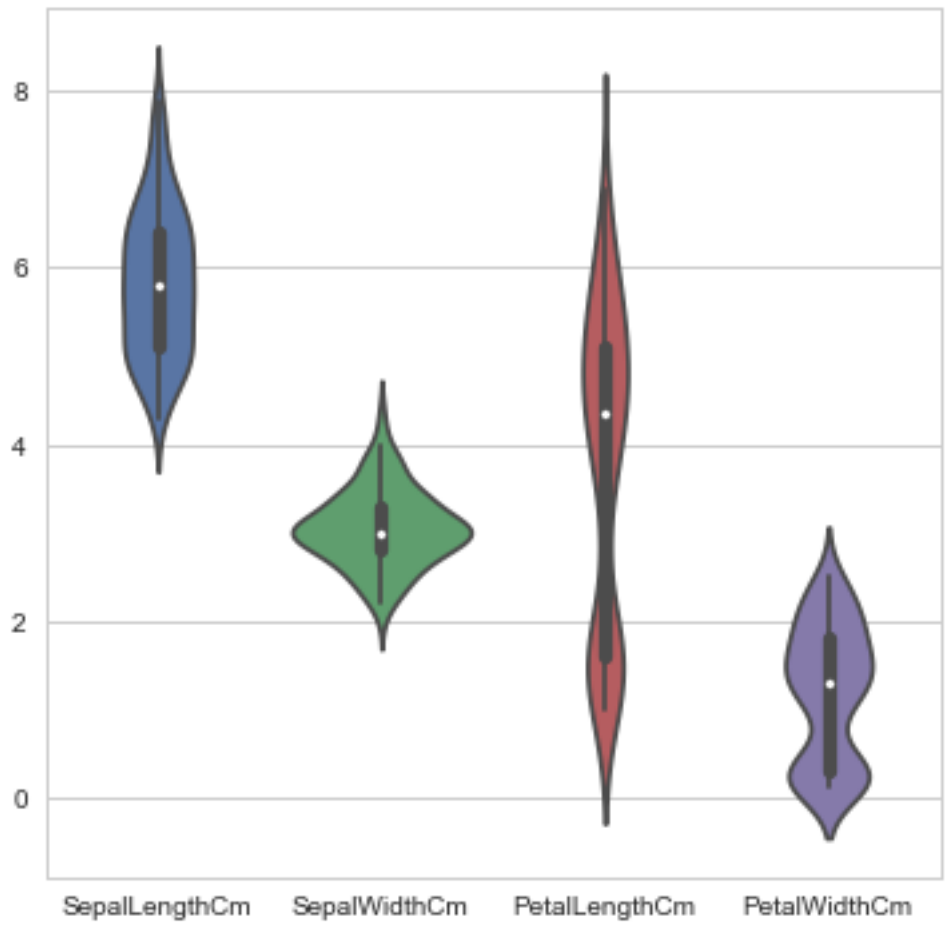
```
# palette = 'Set2' pour changer les couleurs
ax = sns.boxplot(data=df, palette='Set2')
```



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

In [20]:

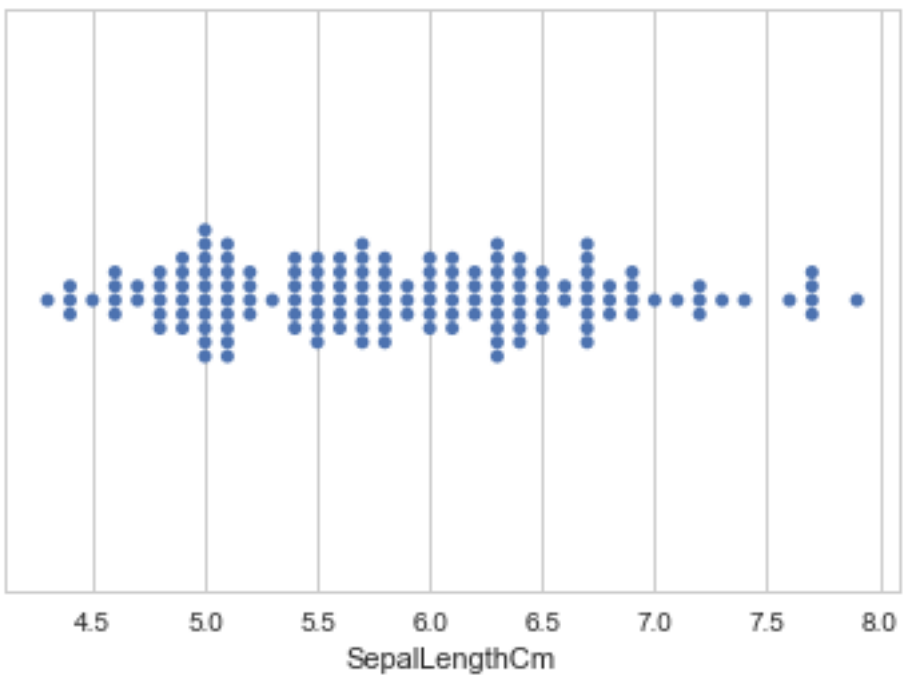
```
# changer la taille de la figure
fig, ax = plt.subplots(figsize=(6,6))
# Affichage en violon
ax=sns.violinplot(data=df)
```



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 [21]:

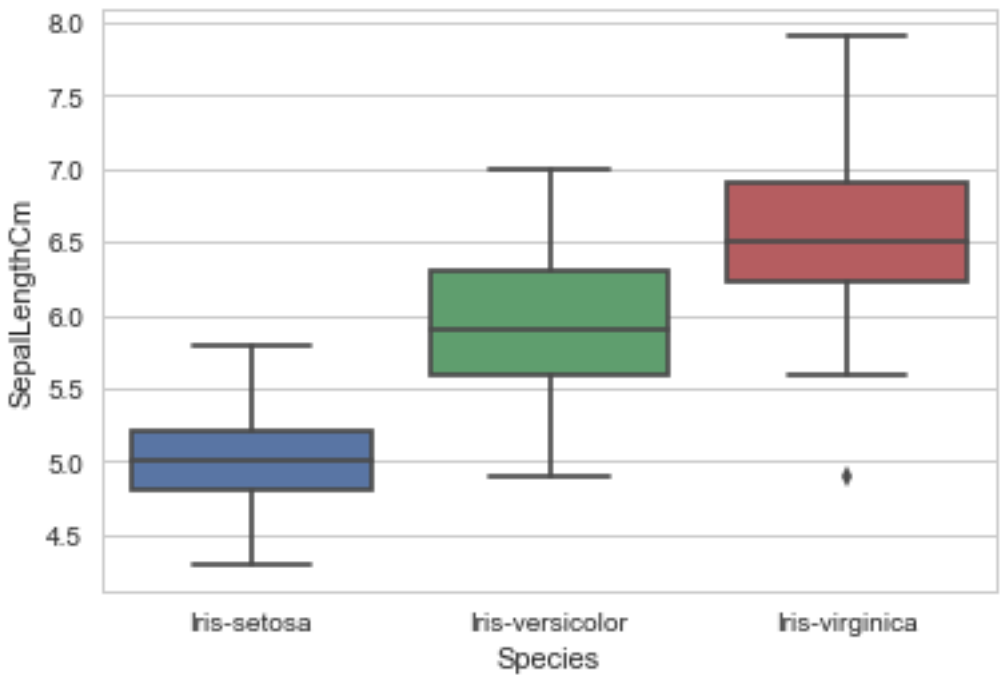
```
ax = sns.swarmplot(x=df["SepalLengthCm"])
```



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

In [22]:

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



Travailler avec plusieurs attributs

In [23]:

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

Out[23]:

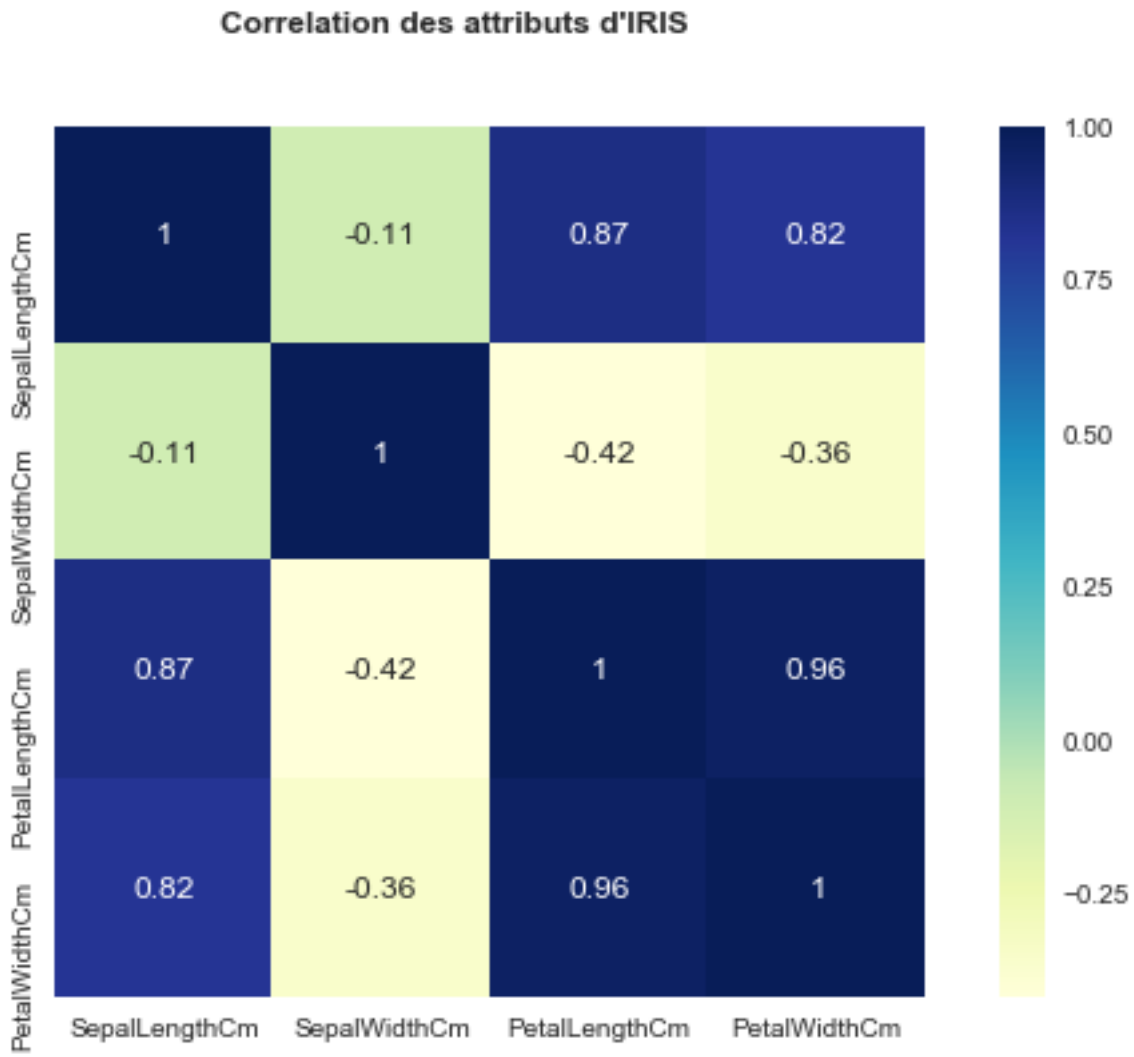
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [24]:

```
fig, ax = plt.subplots(1, 1, figsize=(10,6))

hm = sns.heatmap(corr,
                 ax=ax,           # Axes où afficher
                 cmap="YlGnBu", # Couleur
                 square=True,    # Si True, toutes les cellules ont le même
                 annot=True      # Pour afficher les valeurs
                )

fig.suptitle("Correlation des attributs d'IRIS",
             fontsize=12,
             fontweight='bold')
```

Out[24]:

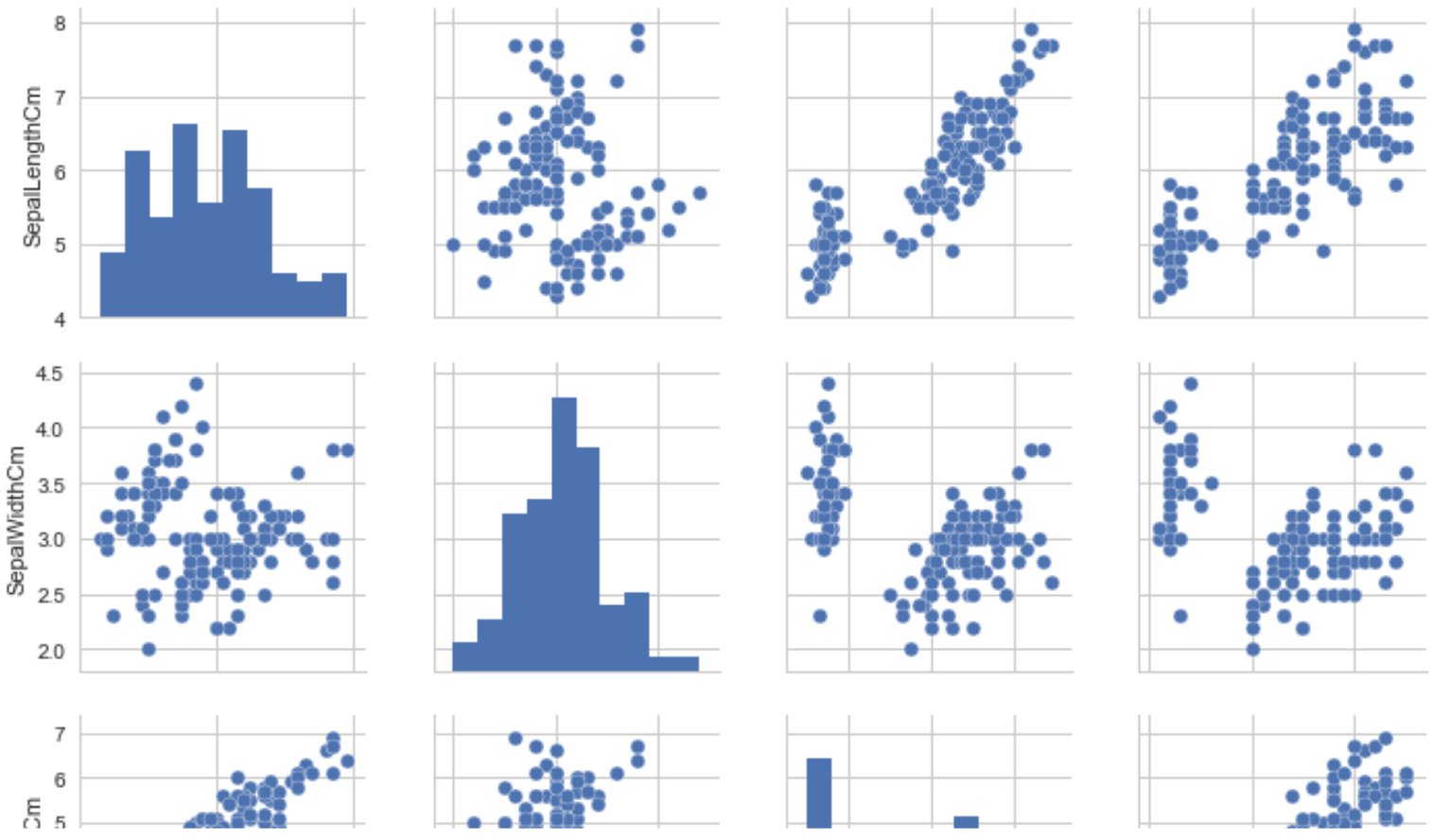
```
Text(0.5, 0.98, "Correlation des attributs d'IRIS")
```



Scatter plot matrix

In [25]:

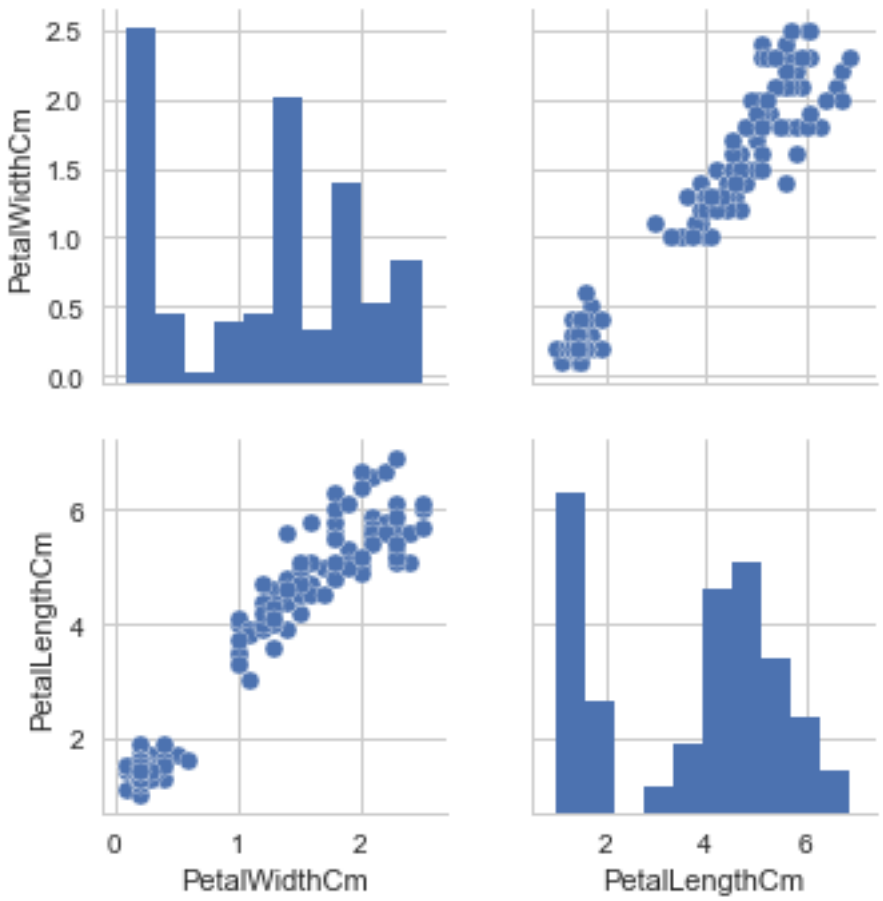
```
# Matrice de base
g = sns.pairplot(df)
```



In [26]:

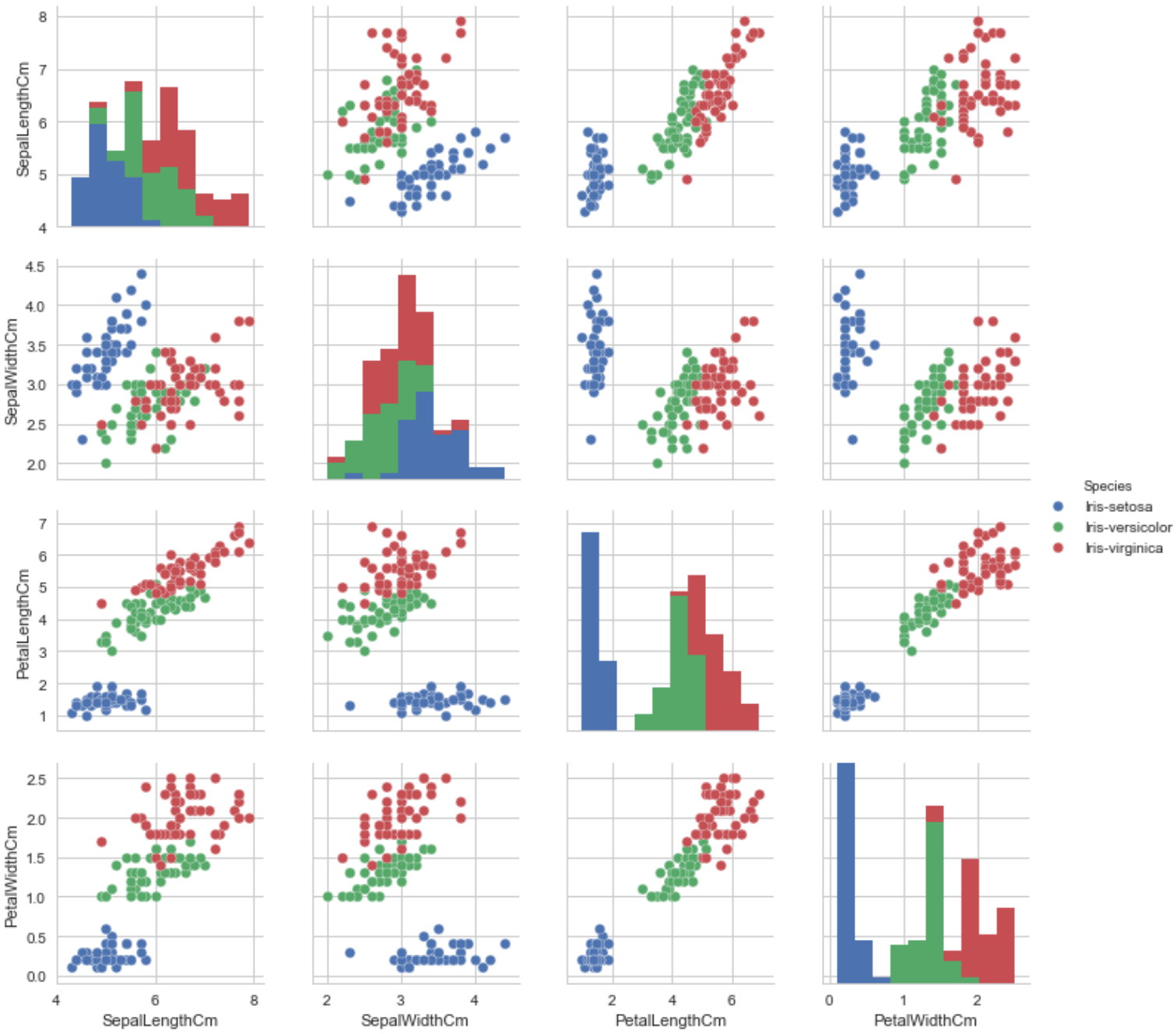
```
# Matrice réduite a un sous ensemble d'attributs
print ("Un exemple de variables fortements corrélées")
g = sns.pairplot(df, vars=["PetalWidthCm", "PetalLengthCm"])


```

```
Un exemple de variables fortements corrélées
```



In [27]:

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



In [28]:

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