# Visualization in Python VS R

Shukai Wang, Nancy Wang

For people working in Data Science field without statistical background, visualization using R might be unfamiliar to them. We hope this tutorial can help those people better use R. To achieve this goal, we summarized the most common used graphics both in Python and R. Users can have a look at difference of the effect and function of these graphics in Python and R clearly because we use the same datasets for the same types of graphic. We emphasizes on the advantages and disadvantages for both of them when dealing with different problems.

These are the packages used for R: Matplotlib; seaborn; numpy; pandas; sklearnl; holoviews

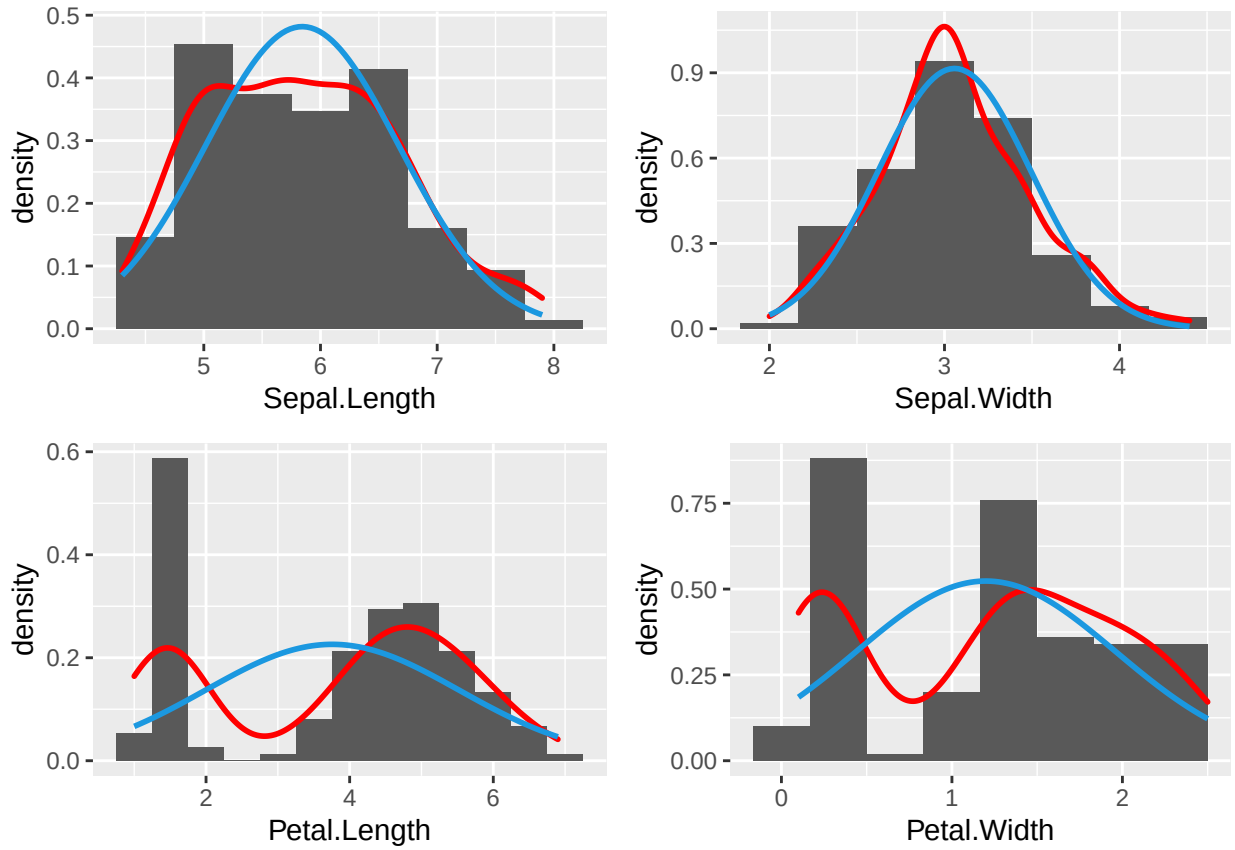
These are the packages used for Python: ggplot2; gridExtra; reshape2; dplyr; vcd; gcookbook; ggalluvial; MASS

## R Visualization

Package ggplot2 offers very strong functions for statistics study. Unlike building a probability density function separately on Python, we just need to use dnorm function in R to build the normal distribution for the hist plots.

```
d1<-ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(aes(y = ..density..),binwidth = 0.5) +
  geom_density(color = 'red',size=1) +
  stat_function(fun = dnorm,
                args = list(mean = mean(iris$Sepal.Length),
                            sd = sd(iris$Sepal.Length)),
                col = "#1b98e0",
                size=1)
d2<-ggplot(data = iris, aes(x = Sepal.Width)) +
  geom_histogram(aes(y = ..density..),binwidth = 1/3) +
  geom_density(color = 'red',size=1) +
  stat_function(fun = dnorm,
                args = list(mean = mean(iris$Sepal.Width),
                            sd = sd(iris$Sepal.Width)),
                col = "#1b98e0",
                size=1)
d3<-ggplot(data = iris, aes(x = Petal.Length)) +
  geom_histogram(aes(y = ..density..),binwidth = 0.5) +
  geom_density(color = 'red',size=1) +
  stat_function(fun = dnorm,
                args = list(mean = mean(iris$Petal.Length),
                            sd = sd(iris$Petal.Length)),
                col = "#1b98e0",
                size=1)
d4<-ggplot(data = iris, aes(x = Petal.Width)) +
  geom_histogram(aes(y = ..density..),binwidth = 1/3) +
  geom_density(color = 'red',size=1) +
```

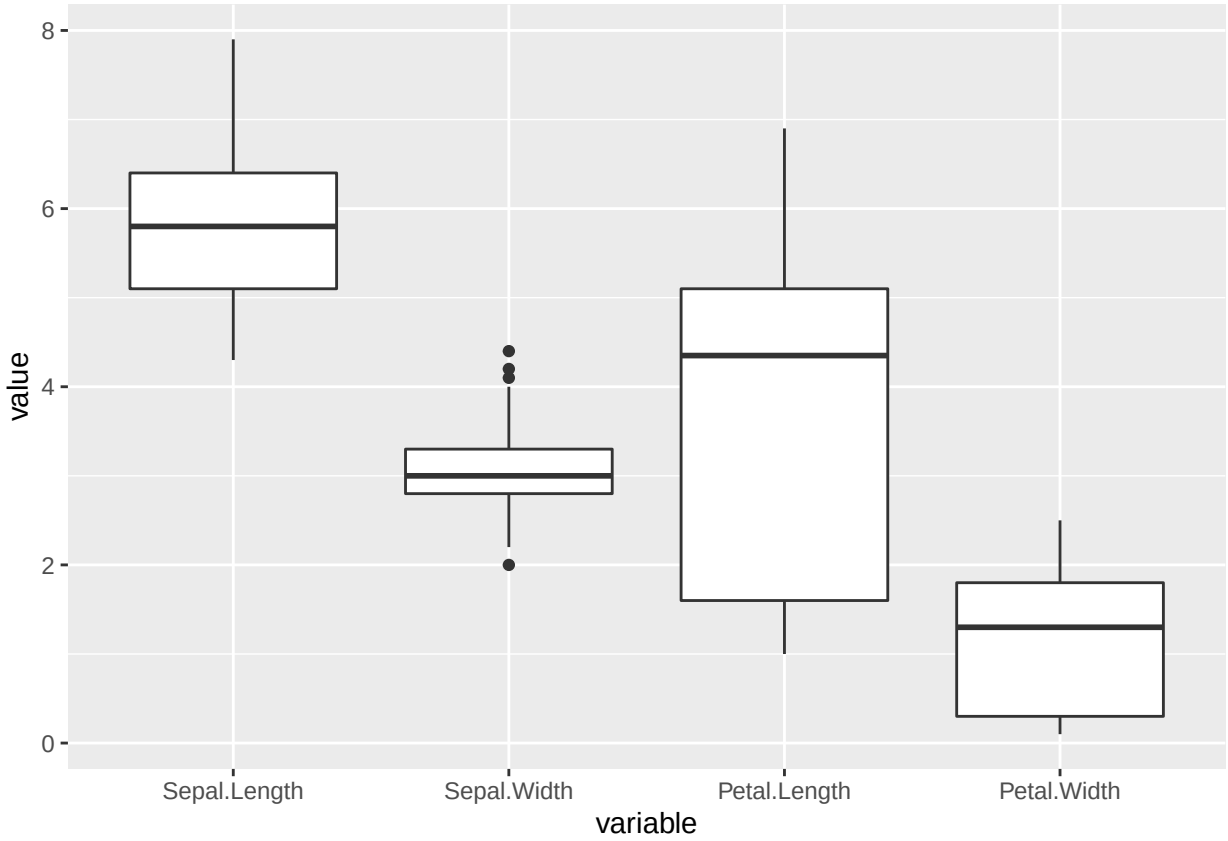
```
  stat_function(fun = dnorm,
                args = list(mean = mean(iris$Petal.Width),
                            sd = sd(iris$Petal.Width)),
                col = "#1b98e0",
                size=1)
grid.arrange(d1,d2,d3,d4,nrow=2)
```



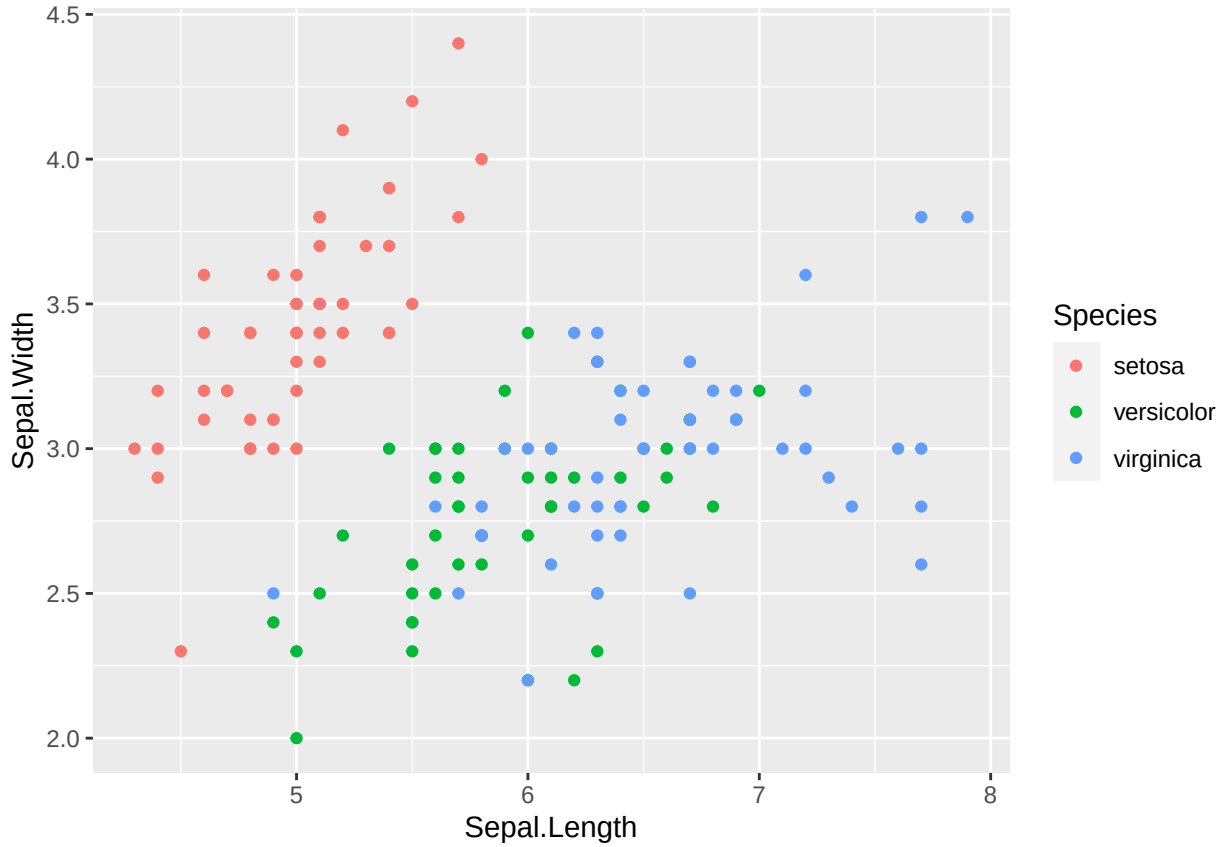

We can directly draw a box plot for one feature. To draw for a few features together and show in one plot, we need the package reshape2 and its melt function, which is quite different from Python where we can input the whole dataframe directly and get the same result.

```
ggplot(melt(iris), aes(x = variable, y = value)) +
  geom_boxplot()
```

```
## Using Species as id variables
```




```
ggplot(data = iris, mapping = aes(x = Sepal.Length, y = Sepal.Width, color=Species)) +
  geom_point()
```

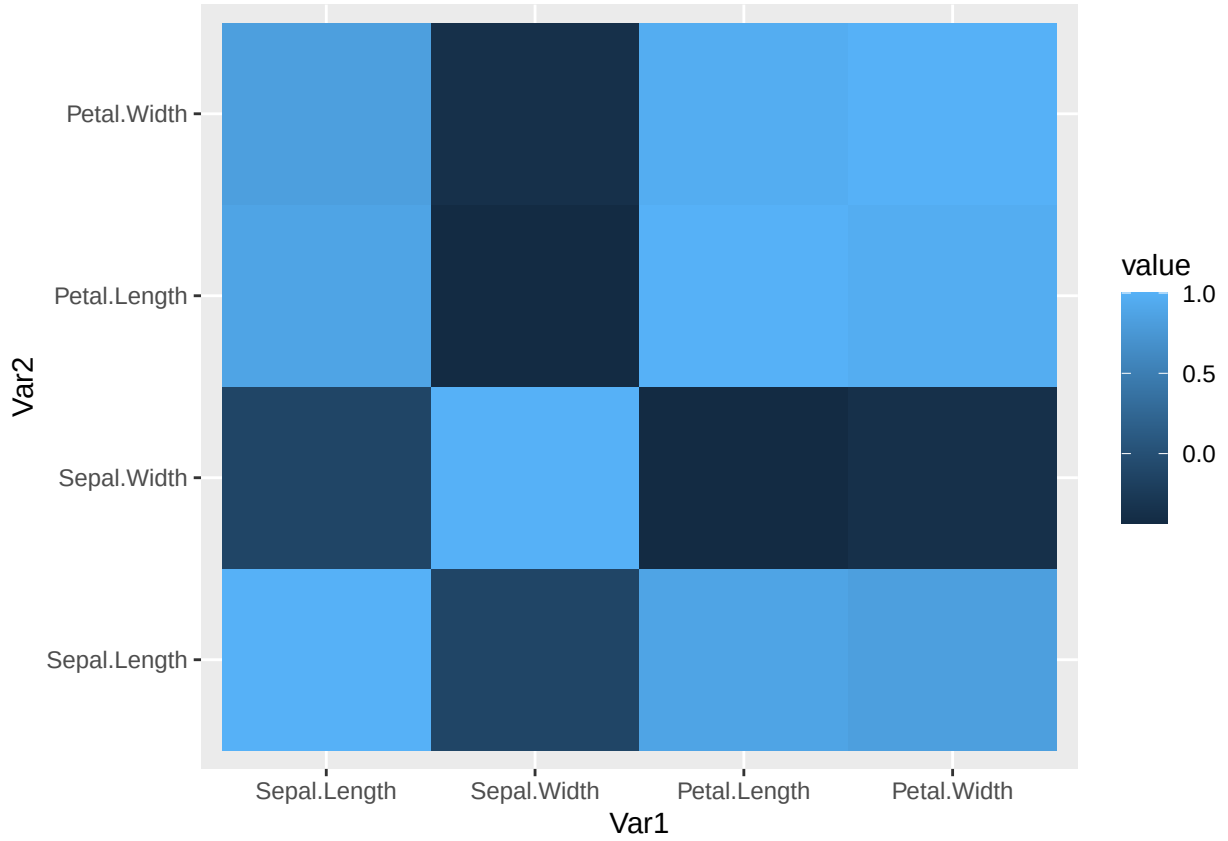



The correlation of the features is an important index to analyze the multicollinearity of variables.To draw a heatmap to represent the correlation of all the features, the procedure based on ggplot2 is little more complex than using Python. We need to specify the value for x-axis, y-axis and filling. We use melt function again here to help us get the result.

```
df <- iris
df <- subset(df, select=-Species)
cormat <- round(cor(df),2)
head(cormat)
```

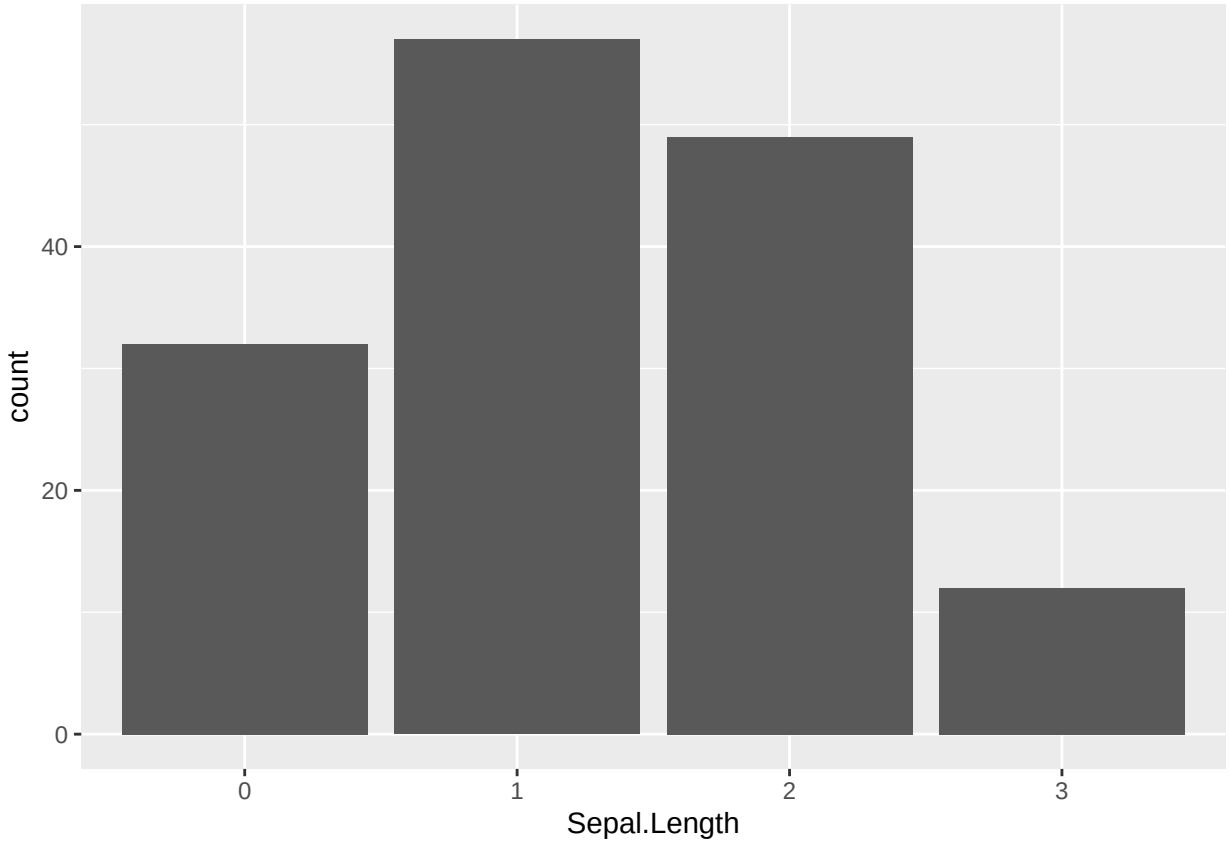
```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length         1.00       -0.12         0.87        0.82
## Sepal.Width         -0.12        1.00        -0.43       -0.37
## Petal.Length         0.87       -0.43         1.00        0.96
## Petal.Width          0.82       -0.37         0.96        1.00
```

```
melted_cormat <- melt(cormat)
#options(repr.plot.width=3,repr.plot.height=3)
ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()
```



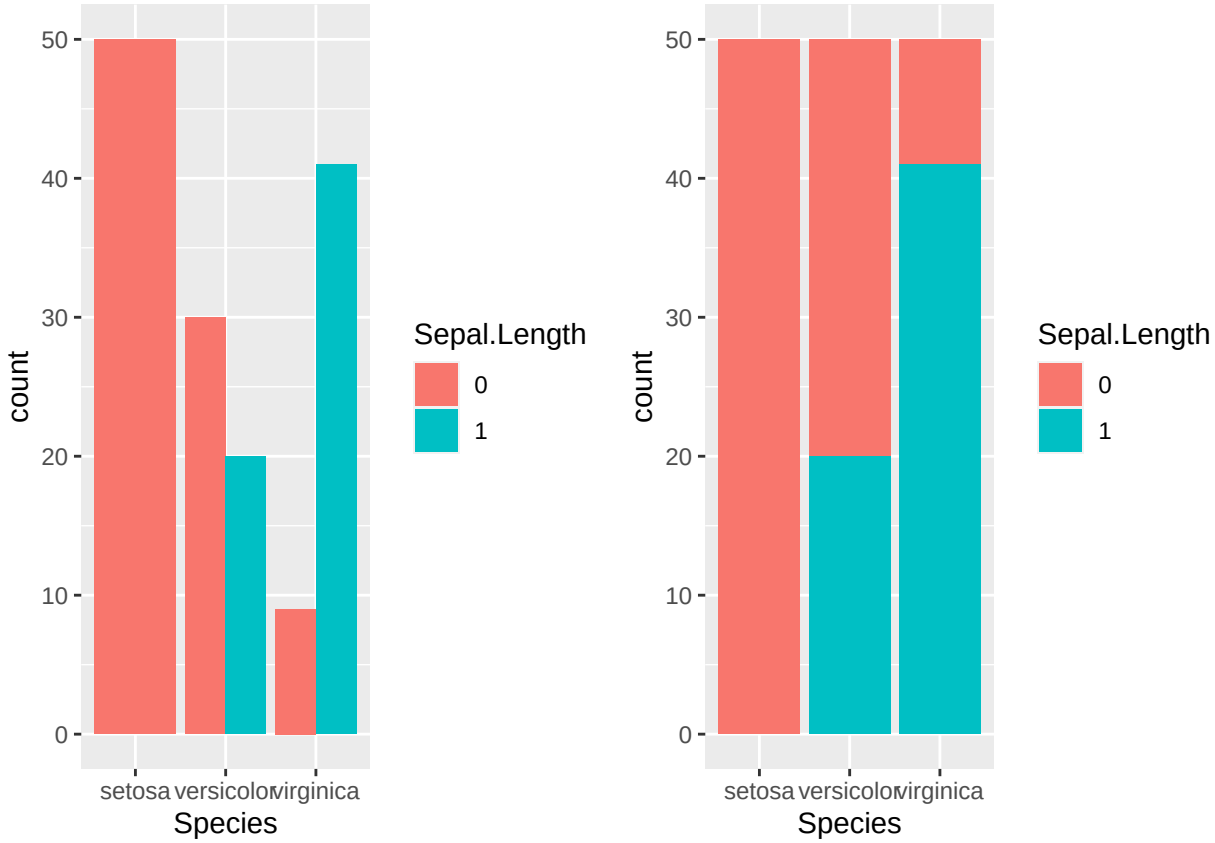

On Python, we need to specify the variable for y-axis, otherwise the programming will has error. While here ggplot2 counts the number of variable for x-axis and does not allow to add another variable for y-axis. We should be careful about the difference when using bar plot on Python or R. Since the counts of each species are the same, we categorize the value of SepalLength to get the plot.

```
df <- iris
df <- df %>% mutate(Sepal.Length=case_when(between(Sepal.Length,4,5) ~ '0',
                                           between(Sepal.Length,5,6) ~ '1',
                                           between(Sepal.Length,6,7) ~ '2',
                                           between(Sepal.Length,7,8) ~ '3',
                                           ))
ggplot(df, aes(x = Sepal.Length)) +
    geom_bar()
```




It is more convenient to implement stacked barplot and grouped barplot on R compared with Python. We categorize the value of SepalLength to get the plot.

```
df <- iris
df <- df %>% mutate(Sepal.Length=case_when(between(Sepal.Length,4,6) ~ '0',
                                           between(Sepal.Length,6,8) ~ '1'))
group <- ggplot(df, aes(x = Species, fill = Sepal.Length)) +
    geom_bar(position = "dodge")
stack <- ggplot(df, aes(x = Species, fill = Sepal.Length)) +
    geom_bar()
grid.arrange(group,stack,nrow=1)
```



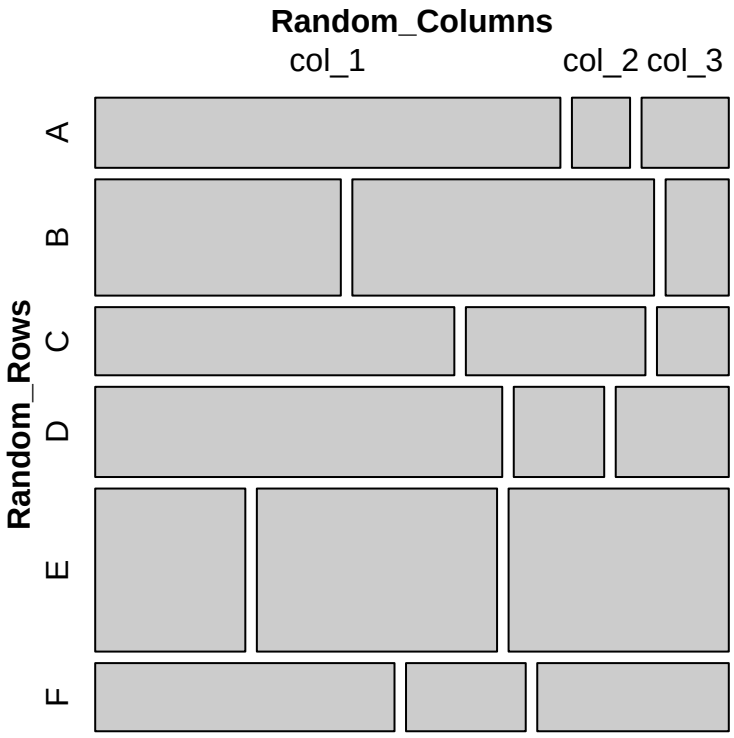

From these two plots we can clearly see the relationship between species and the length of sepals. The lengths of Setosa's sepal are all less than 6.

We randomly generated a categorical data matrix to illustrate how we build a mosaic plot, which is used to show the proportions of categorical variables.

```
#mosaic plot
# creating a random dataset
# creating 6 rows
data_values <- matrix(c(80, 10, 15,
                        70, 86, 18,
                        60, 30, 12,
                        90, 20, 25,
                        60, 96, 88,
                        50, 20, 32))
```

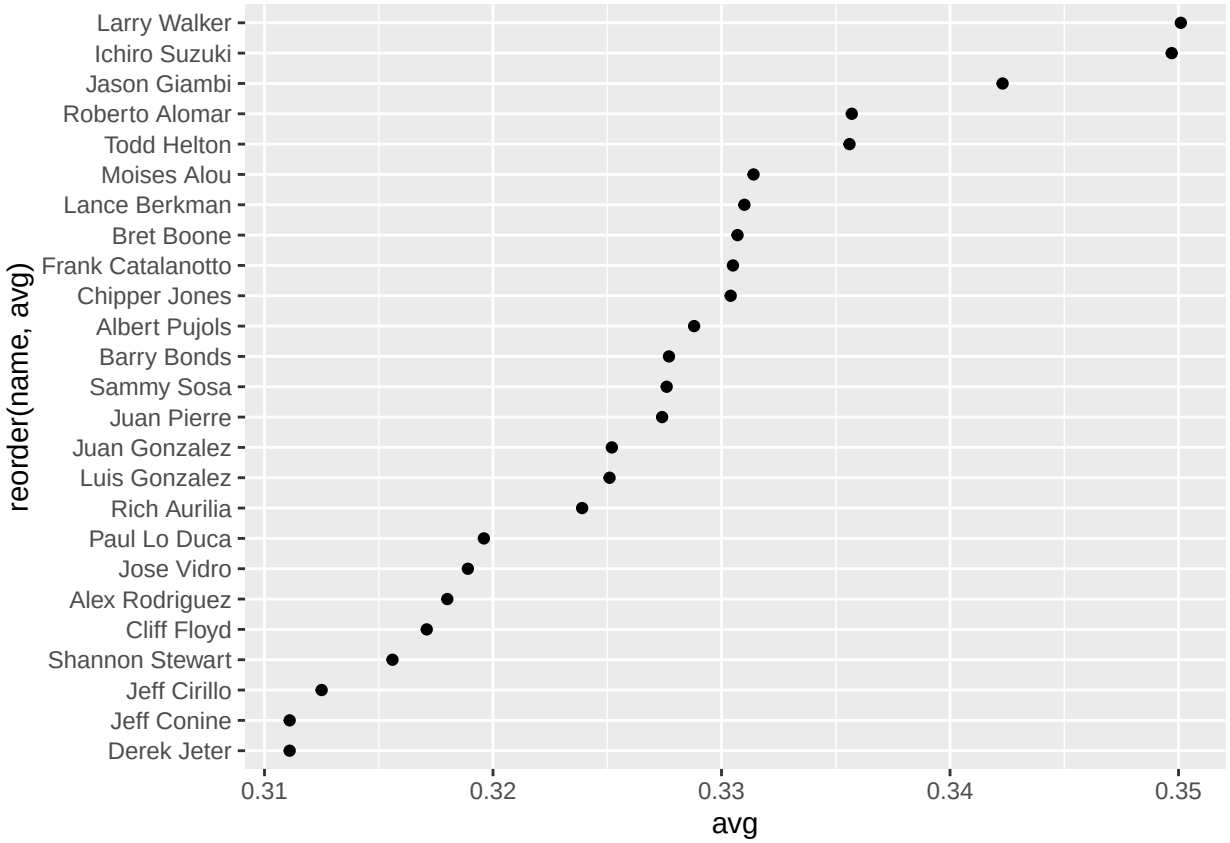
```
data <- as.table(
  matrix(
    data_values,

    # specifying the number of rows
    nrow = 6,
    byrow = TRUE,
    #create variable names
    dimnames = list(
      Random_Rows = c('A','B','C', 'D', 'E', 'F'),
      Random_Columns = c('col_1', 'col_2', 'col_3')
```

```
    )
  )
)
mosaic(data)
```



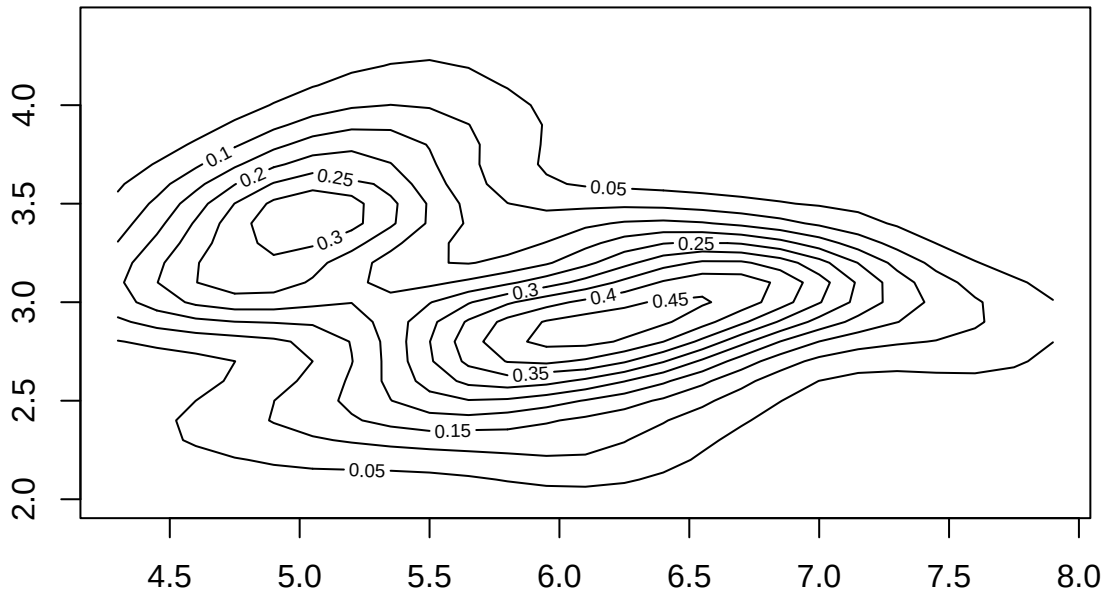

We use the tophitters2001 data set to implement the cleverland dot plot.

```
#Load gcookbook for the tophitters2001 data set
tophit <- tophitters2001[1:25, ]
ggplot(tophit, aes(x = avg, y = reorder(name, avg))) +
  geom_point()
```



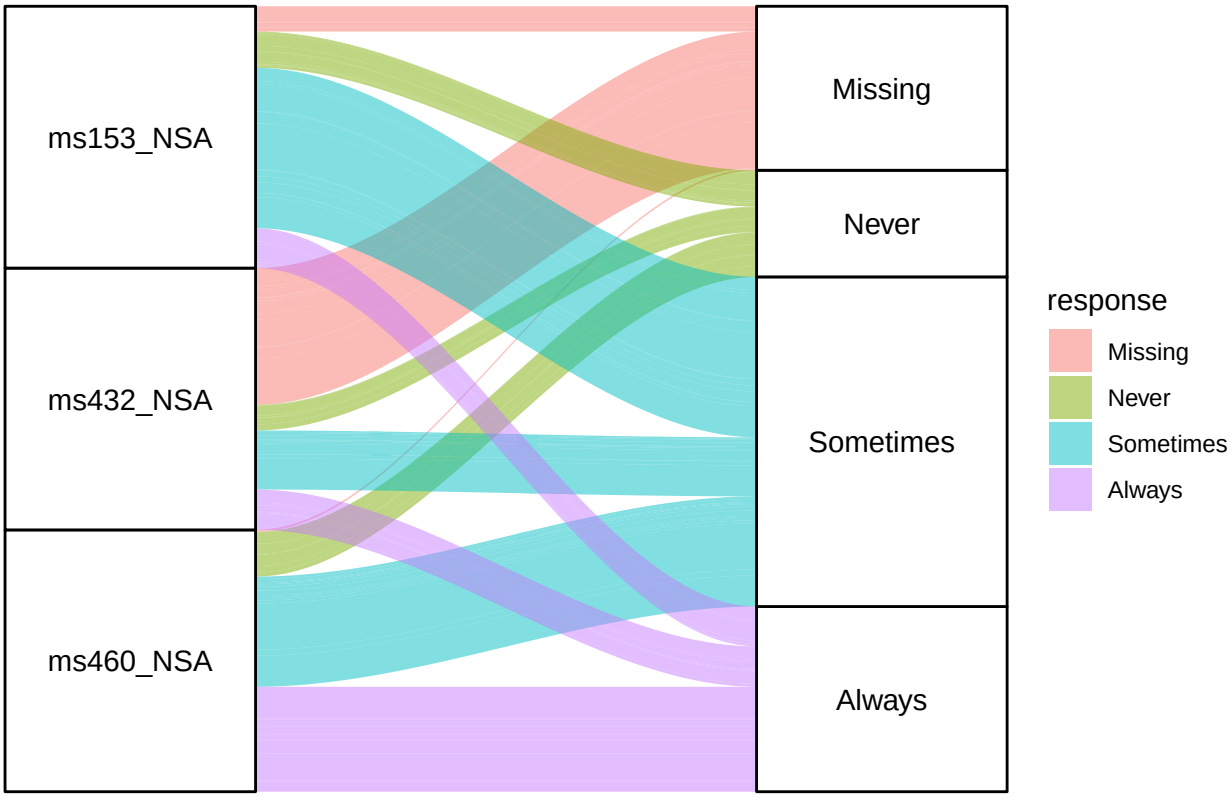

We use iris datset to build a contour plot. The results clear shows the ranges of variables.

```
#contour plot
x <- as.numeric(as.character(iris$Sepal.Length))
y <- as.numeric(as.character(iris$Sepal.Width))
z <- kde2d(x, y)
contour(z)
```




Alluvial plot is another plot method to reveal the relationship between categorical variables. We use the vaccinations dataset in r.ggalluvial package to illustrate it.

```
ggplot(data = vaccinations,
       aes(axis1 = survey, axis2 = response, y = freq)) +
  geom_alluvium(aes(fill = response),
                curve_type = "cubic") +
  geom_stratum() +
  geom_text(stat = "stratum",
            aes(label = after_stat(stratum))) +
  scale_x_discrete(limits = c("Survey", "Response"),
                   expand = c(0.15, 0.05)) +
  theme_void()
```


ms153_NSA
ms432_NSA
ms460_NSA
Missing
Never
Sometimes
Always
response
Missing
Never
Sometimes
Always

# Python Visualization

November 15, 2022

## 1 Python Visualization

[1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

We load iris data from packge sklearn. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Both Python and R has the access to this dataset.

[2]:
```
from sklearn.datasets import load_iris
iris = load_iris()
iris
df = pd.DataFrame(iris['data'], columns=['SepalLength', 'SepalWidth',␣
↪'PetalLength', 'PetalWidth'])
cg = pd.DataFrame(iris['target'],columns=['Category'])
```

It is not as convenient to build the normal distribution of histplot in Python as in R. We need to build a function to construct the probability distribution function and use the function "lineplot" form seaborn to draw the line.

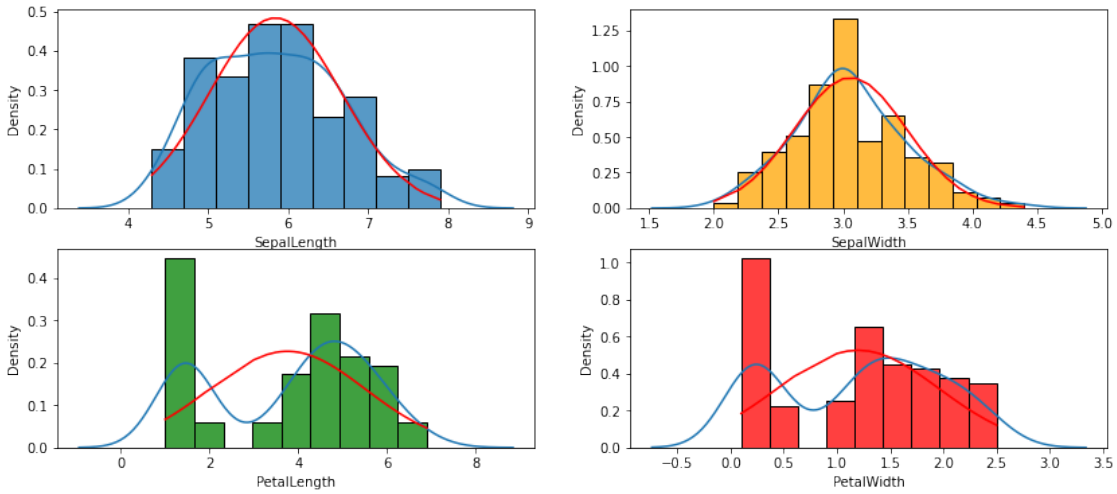
[3]:
```
def normfun(x):
    mu = np.mean(x)
    sigma = np.std(x)
    pdf = np.exp(-((x - mu) ** 2) / (2 * sigma ** 2)) / (sigma * np.sqrt(2 * np.
↪pi))
    return pdf
```

The distribution of one or more variables is represented by a histogram, a traditional visualization tool, by counting the number of data that fall within discrete bins. The distribution of observations in a dataset can be seen visually using a kernel density estimate (KDE) plot, which represents the data as a continuous probability density curve in one or more dimensions.

[4]:
```
fig,ax = plt.subplots(nrows=2,ncols=2,figsize=(14,6))
axes = ax.flatten()
sns.histplot(data=df,x='SepalLength',stat='density',ax=axes[0])
sns.kdeplot(data=df,x='SepalLength',ax=axes[0])
```

```
sns.lineplot(data=df,x='SepalLength',y=normfun(df.
 ↪SepalLength),ax=axes[0],color='red')
sns.histplot(data=df,x='SepalWidth',stat='density',color='orange',ax=axes[1])
sns.kdeplot(data=df,x='SepalWidth',ax=axes[1])
sns.lineplot(data=df,x='SepalWidth',y=normfun(df.
 ↪SepalWidth),ax=axes[1],color='red')
sns.histplot(data=df,x='PetalLength',stat='density',color='green',ax=axes[2])
sns.kdeplot(data=df,x='PetalLength',ax=axes[2])
sns.lineplot(data=df,x='PetalLength',y=normfun(df.
 ↪PetalLength),ax=axes[2],color='red')
sns.histplot(data=df,x='PetalWidth',stat='density',color='red',ax=axes[3])
sns.kdeplot(data=df,x='PetalWidth',ax=axes[3])
sns.lineplot(data=df,x='PetalWidth',y=normfun(df.
 ↪PetalWidth),ax=axes[3],color='red')
```

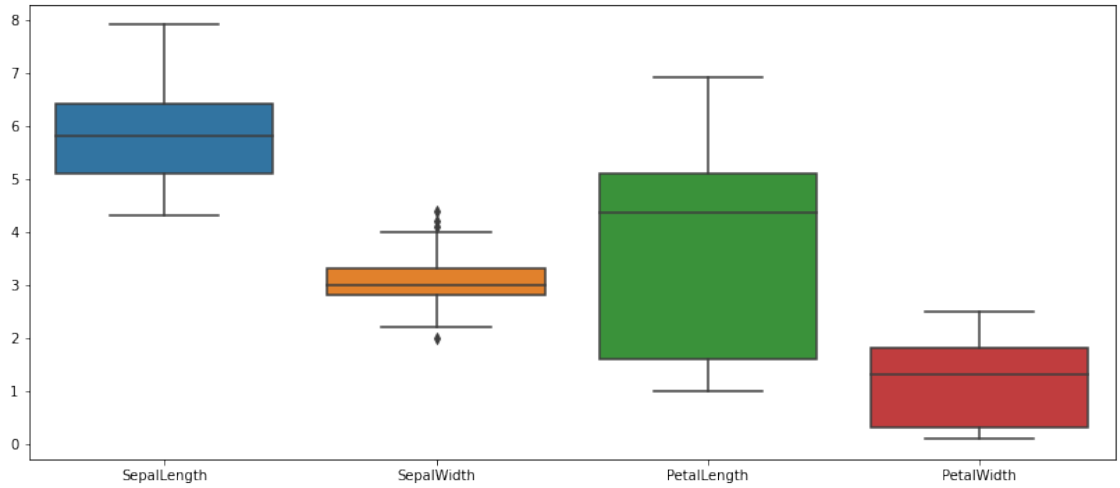
[4]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f3914901410>`




In order to compare variables or levels of a categorical variable, a box plot illustrates the distribution of quantitative data. Except for points that are identified as "outliers", the box plot displays the dataset's quartiles, and the whiskers expand to display the remainder of the distribution.

[5]:
```
plt.figure(figsize=(14,6))
sns.boxplot(data=df)
```

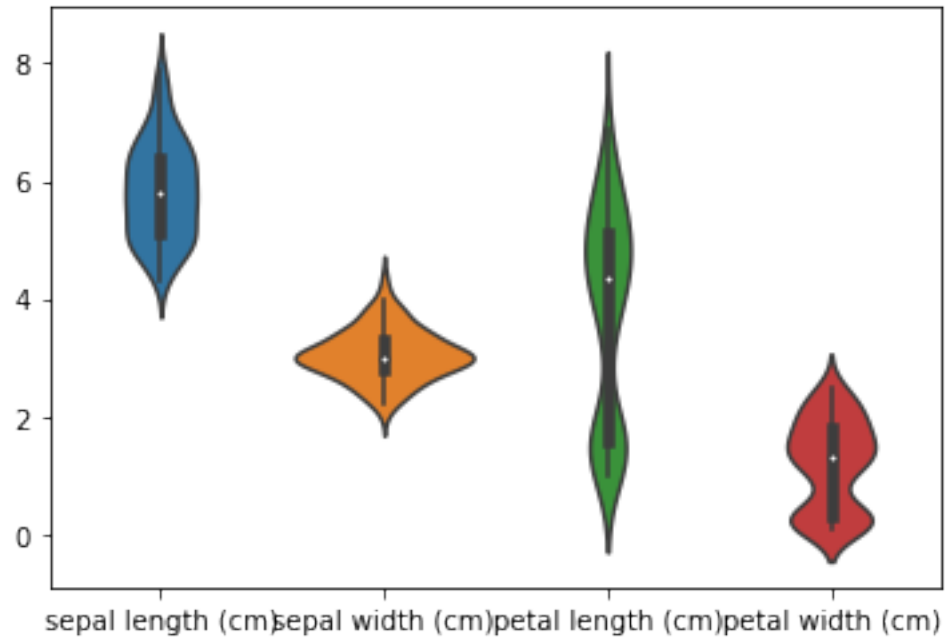
[5]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f3914654910>`




similar with boxplot, violin plot is another method to compare variables or levels of a categorical variable.

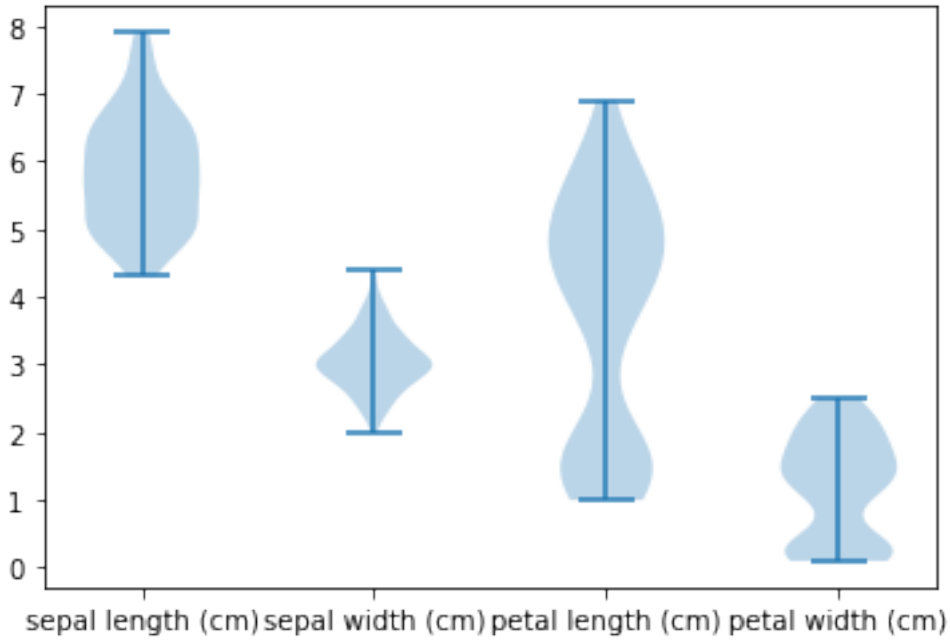
[6]:
```
g1 = sns.violinplot(data = iris.data)
g1.set_xticklabels(iris.feature_names)
```

[6]:
```
[Text(0, 0, 'sepal length (cm)'),
 Text(0, 0, 'sepal width (cm)'),
 Text(0, 0, 'petal length (cm)'),
 Text(0, 0, 'petal width (cm)')]
```



[7]:
```
#violplot using matplotlib
fig, ax = plt.subplots()
ax.violinplot(dataset = iris.data)
ax.set_xticks([1,2,3,4])
ax.set_xticklabels(iris.feature_names)
```

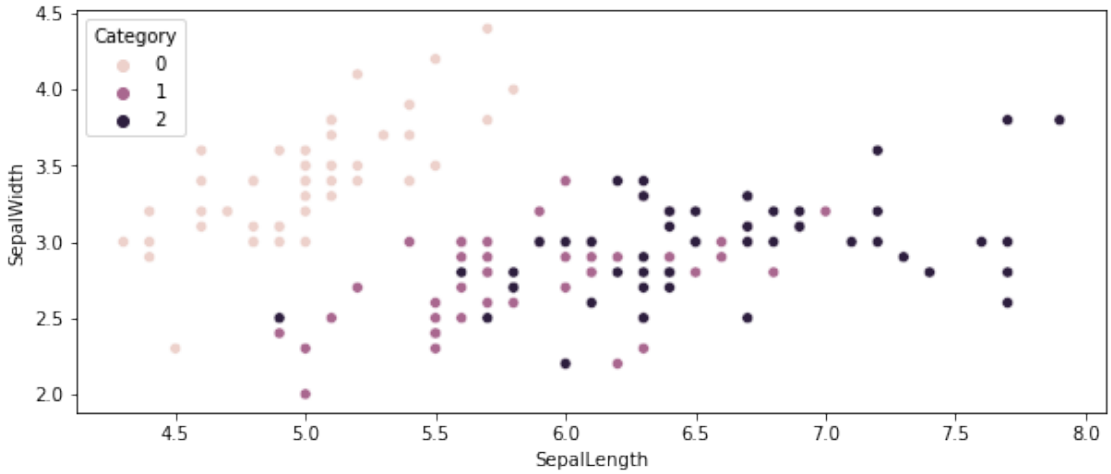
[7]:
```
[Text(0, 0, 'sepal length (cm)'),
 Text(0, 0, 'sepal width (cm)'),
 Text(0, 0, 'petal length (cm)'),
 Text(0, 0, 'petal width (cm)')]
```




Scatter plot is most used for generally find a relationship between two numerical variables. The relationship between x and y can be shown for different subsets of the data using the hue, size, and style parameters.

[8]:
```
plt.figure(figsize=(10,4))
sns.scatterplot(data=df,x='SepalLength',y='SepalWidth',hue=cg.Category)
```

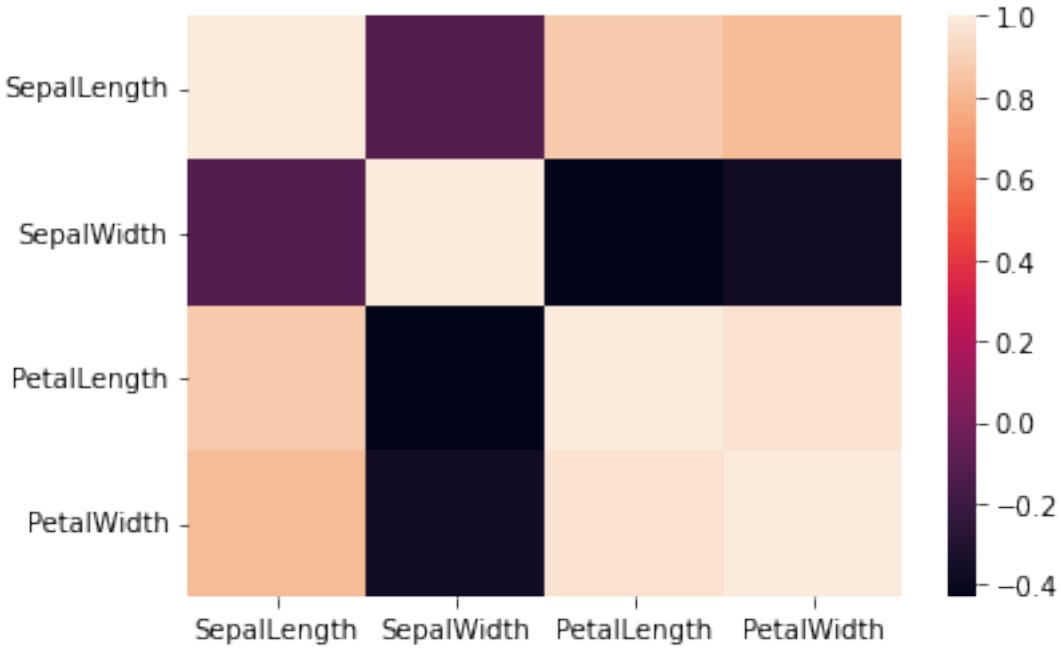
[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f39140dbc10>
```




Heatmap can be used to analyze the relationship within the features. We can compute the correlation first and draw the plot. This is an Axes-level function

[9]:
```
corr = df.corr()
sns.heatmap(corr)
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3913f86b90>

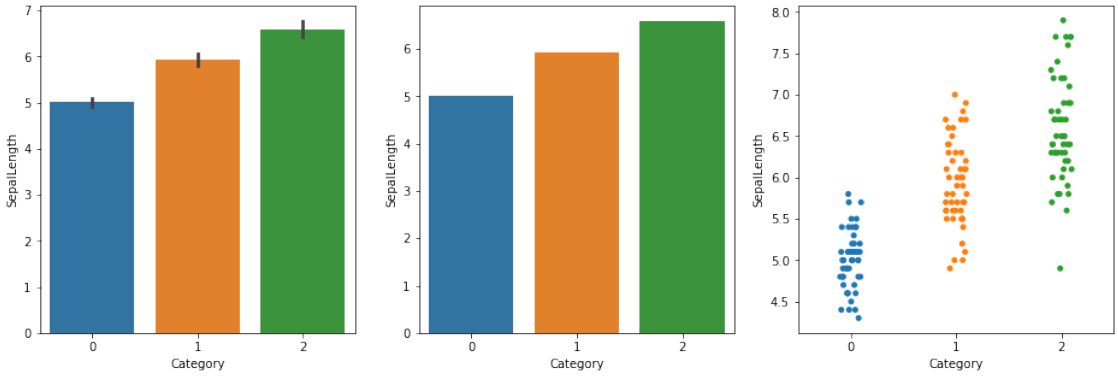



A bar plot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error

bars. Bar graph can also be used to show multiple variables, the function uses the average value as default.

[10]:
```
iris_df = pd.concat([df,cg],axis=1)
mean=iris_df.groupby('Category')['SepalLength'].mean().reset_index()
fig,(ax1,ax2,ax3) = plt.subplots(nrows=1,ncols=3,figsize=(16,5))
sns.barplot(data=iris_df,x='Category',y='SepalLength',ax=ax1)
sns.barplot(data=mean, x="Category", y="SepalLength",ax=ax2)
sns.stripplot(data=iris_df,x='Category',y='SepalLength',ax=ax3)
```

[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f39116bf090>
```

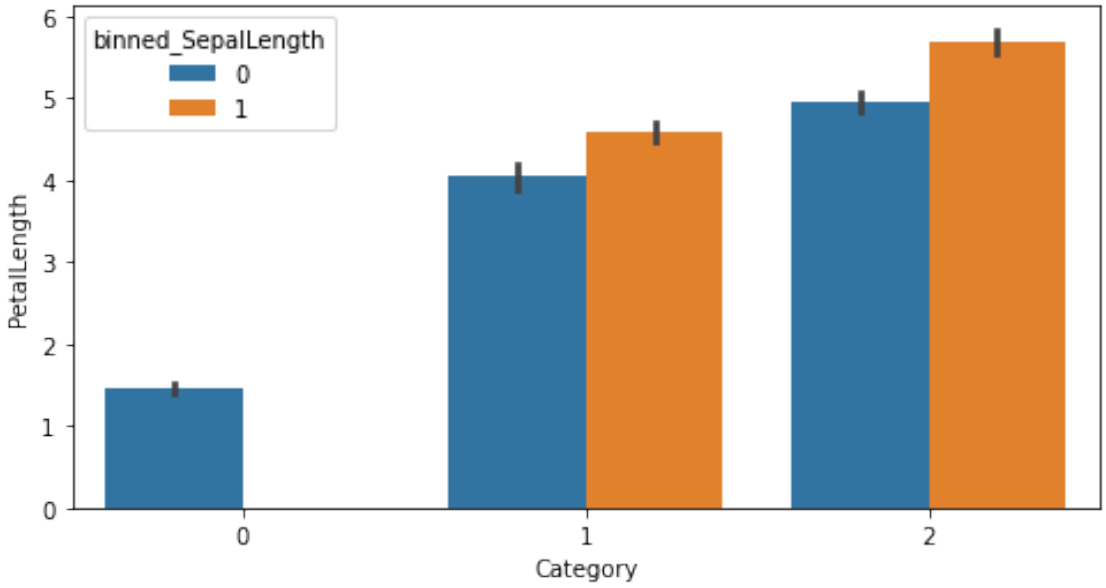



When we have more than one categorical variables, we can use stacked or grouped bar plot. To make the tutorial easy to understand, we binned the features of iris and look at the result. The use of stacked bar plot on Python is not very convenient so we didn't mention it here.

[11]:
```
bins=[4,6,8]
labels=np.arange(0,len(bins)-1)
iris_df['binned_SepalLength']=pd.cut(df['SepalLength'],bins,labels=labels)
```

[12]:
```
plt.figure(figsize=(8,4))
sns.barplot(data=iris_df,x='Category',y='PetalLength',hue='binned_SepalLength')
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3911621cd0>
```

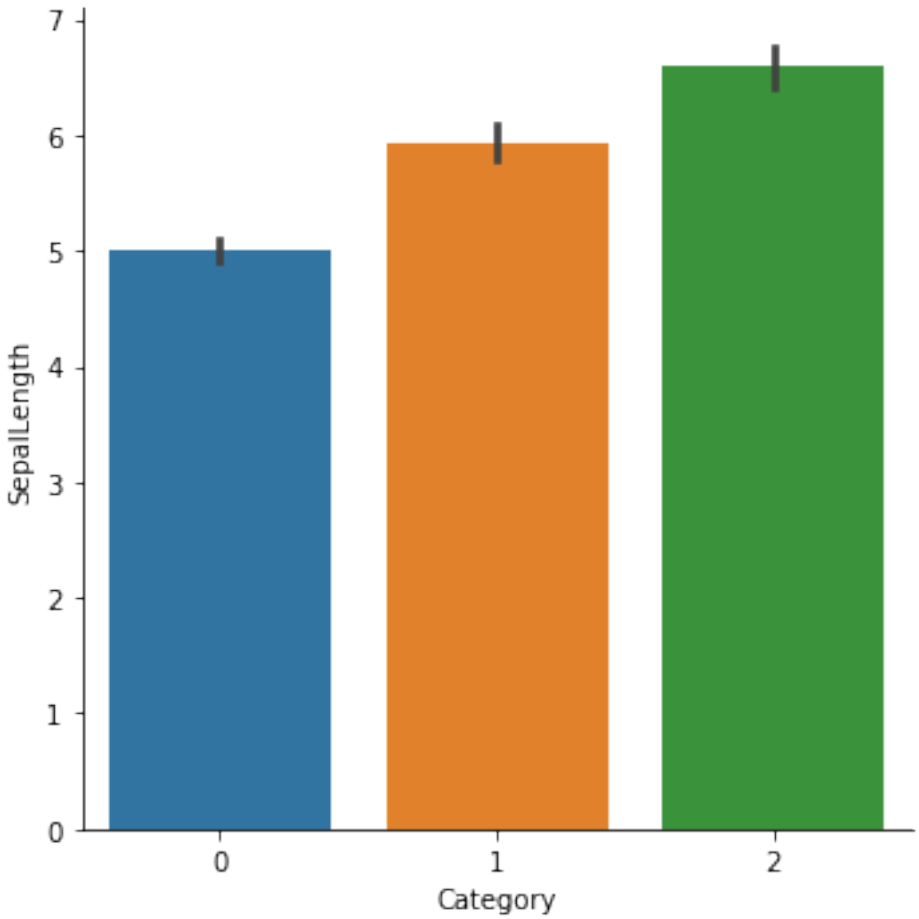



Actually, we can use 'sns.catplot' function to implement any plot with catrgorical variable as x-axis by changing the parameter 'kind'. This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations and the kind parameter selects the underlying axes-level function to use. Take bar plot as an example, it has the same plot as 'sns.barplot'.

Pay attention to the use of subplots here. Catplot is a figure-level function and does not accept target axes.

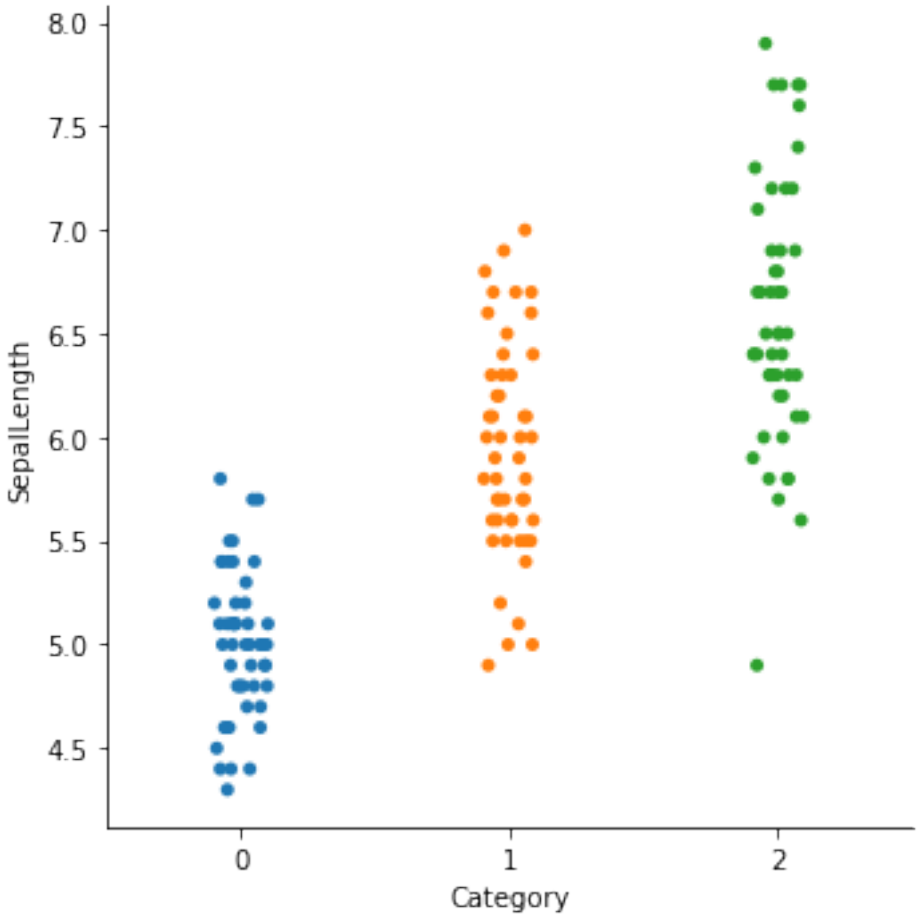
[13]:
```
sns.catplot(data=iris_df,x='Category',y='SepalLength',kind='bar')
```

[13]:
```
<seaborn.axisgrid.FacetGrid at 0x7f39149bb9d0>
```



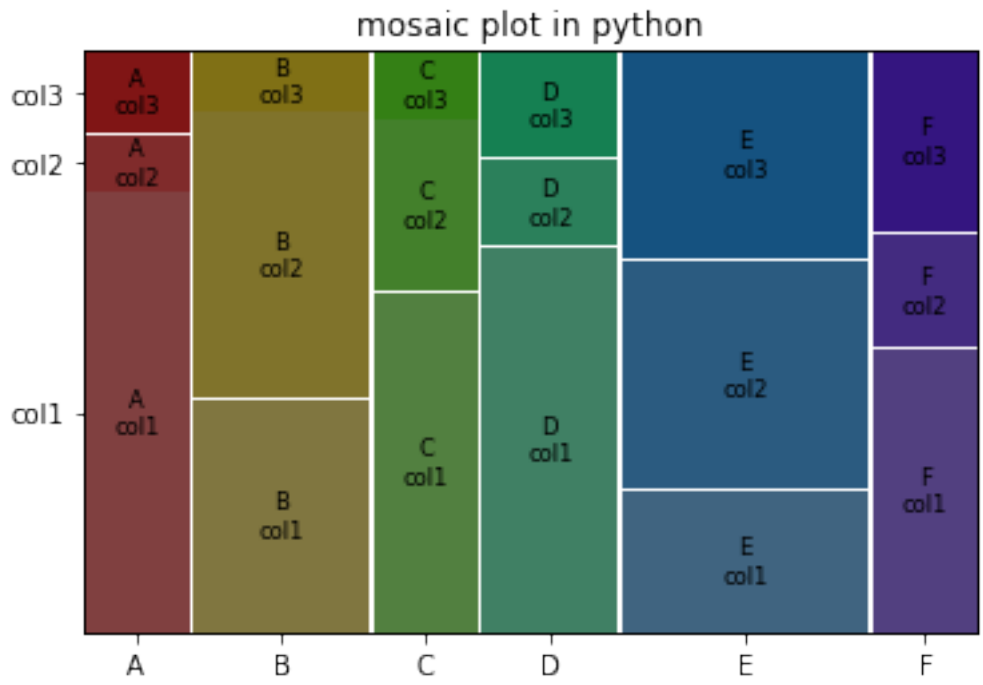

[14]:
```
sns.catplot(data=iris_df,x='Category',y='SepalLength',kind='strip')
```

[14]: <seaborn.axisgrid.FacetGrid at 0x7f39140c5a90>



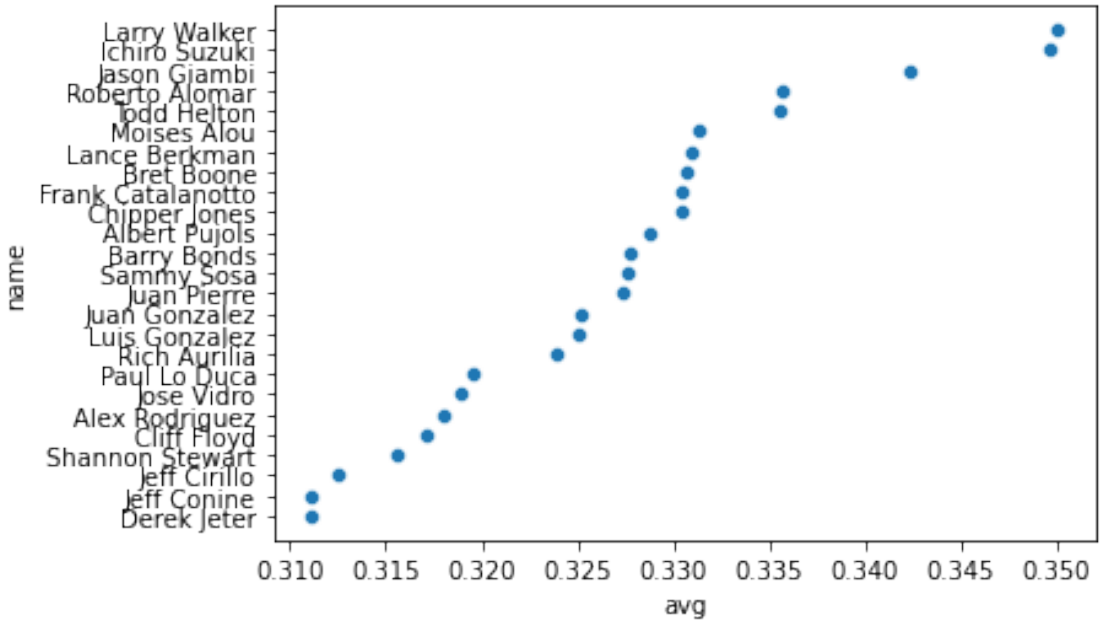

Mosaic plot is a special type of stacked bar chart for categorical variables. We use the statmodels package in python to implement it.(There are not convenient API in matplotlib/seaborn, but using FacetGrid() in matplotlib can also draw a mosaic plot.) The data are created randomly.

[15]:
```
data = [80, 10, 15, 70, 86, 18, 60, 30, 12, 90, 20, 25, 60, 96, 88, 50, 20, 32]
from statsmodels.graphics.mosaicplot import mosaic
from itertools import product
tuples = list(product(["A","B","C","D","E","F"], ['col1', 'col2','col3']))
index = pd.MultiIndex.from_tuples(tuples)
data = pd.Series(data, index=index)
mosaic(data, title='mosaic plot in python')
plt.show()
```




The following is Cleveland dot plot in python. we use part of the tophit dataset in r.gcookbook package.

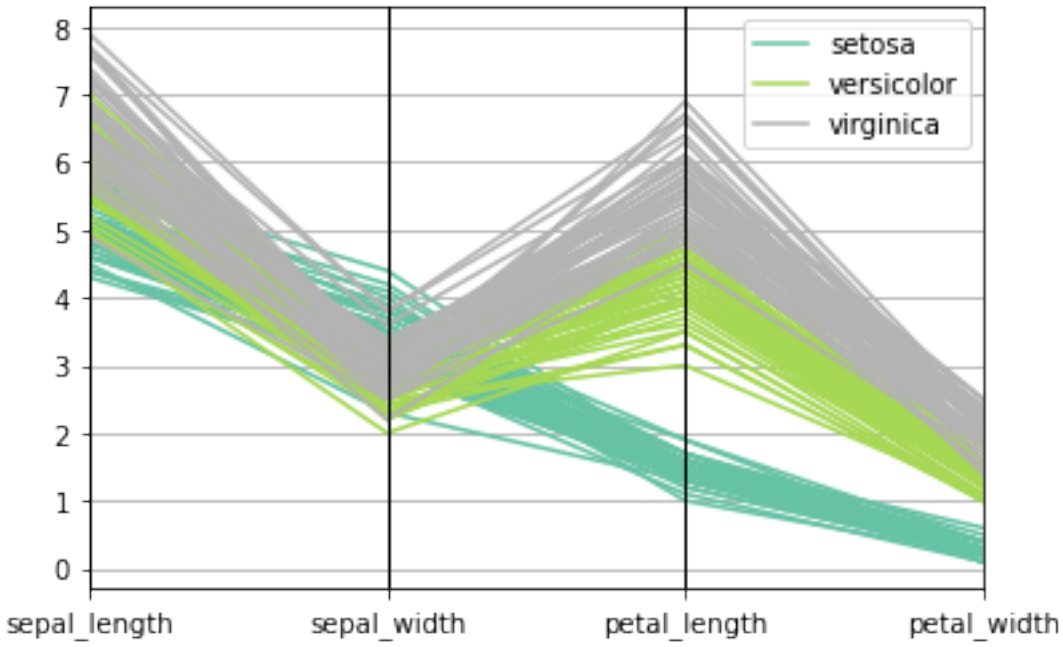
[16]:
```
data = pd.read_csv("tophit.csv")
data = data[:25]
fig = sns.scatterplot(y="name", x="avg", data = data)
```



Parallel coordinates plot are used to show the relationship between different variables. The following plot clearly shows the relationship in iris dataset. We use the API in pandas to implement it.

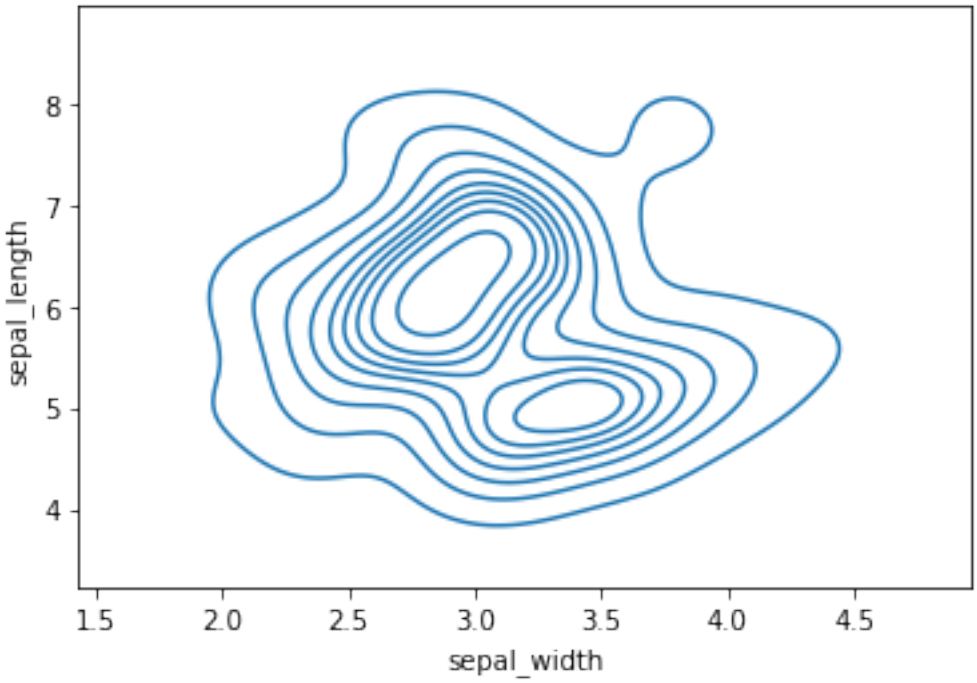
[17]:
```
from pandas.plotting import parallel_coordinates
iris = sns.load_dataset('iris')
parallel_coordinates(iris, 'species', colormap=plt.get_cmap("Set2"))
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3910b37f90>
```




The following is contour plot in python. We still use the iris dataset. Contour plot is a graphical method to visualize the 3-D surface by plotting constant Z slices called contours in a 2-D format.

[21]:
```
sns.kdeplot(x=iris.sepal_width, y=iris.sepal_length)
plt.show()
```




Alluvial plot is another plot method to reveal the relationship between categorical variables. We use the vaccinations dataset in r.ggalluvial package to illustrate it.

[23]:
```
import holoviews as hv
from holoviews import opts, dim
data = pd.read_csv("vaccinations.csv")
hv.extension('bokeh')
sankey = hv.Sankey(data,kdims=["survey", "response"])
sankey.opts(width=600, height=400)
```

[23]: :Sankey [survey,response] (freq,subject,start_date,end_date)

## 2 Reference

https://en.wikipedia.org/wiki/Iris_flower_data_set

http://seaborn.pydata.org/generated/seaborn.catplot.html

https://r-charts.com/distribution/cleveland-dot-plot/

https://holoviews.org/reference_manual/holoviews.element.html?highlight=sankey#holoviews.element.sankey.Sa