Topic 9
Data plotting

Learning Outcomes

After completing this topic and the recommended reading, you should be able to:

- Describe data visualisations and justify their use.
- Select and critically evaluate data visualisations for different types of data, justifying their choice in terms of data communication principles.
- Create data visualisations such as graphs, histograms.

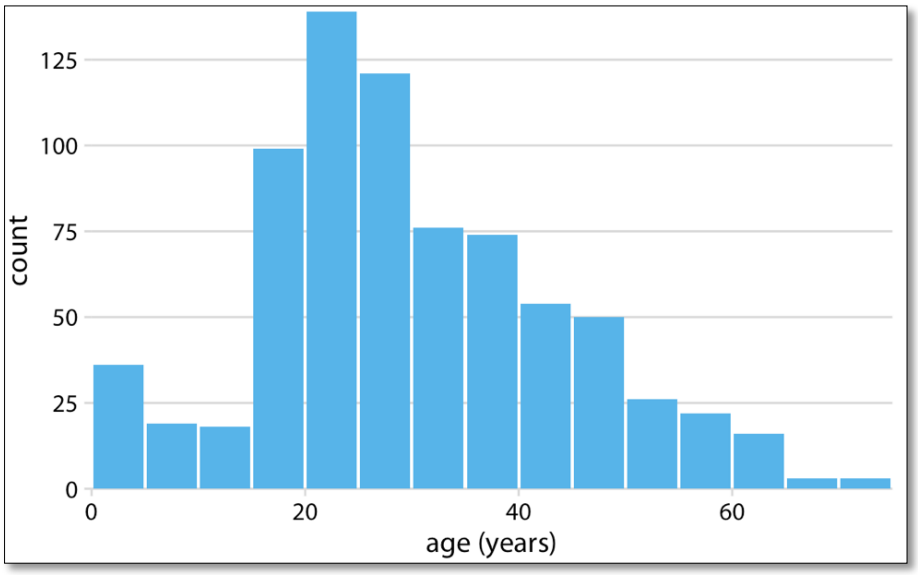
1. Basic Graphs and Principles for Visualising Data

Variables

- Also referred to as ***random variable***.
- Fundamental concept for any type of data analysis.
    - To view the data as the outcome of an experiment/event with uncertainty.
    - To identify patterns in the data.
- ***Continuous variables***:
    - Referred to quantities that can be measured.
    - Assume an infinite number of real values within a given interval.
    - Different levels of precision are possible.
    - Examples:
        - Weight; height; age; etc.
- ***Categorical ordinal variables***:
    - Referred to quantities that take values in pre-specified categories that can be ordered.
    - Example:
        - Educational level: 1. Elementary; 2. High school; etc.
        - Income level: Low; Medium; High.
        - Opinion: 1. Strongly agree; 2. Agree; 3. Neutral; 4. Disagree; 5. Strongly disagree.
- ***Categorical nominal variables***:
    - Referred to named variables without providing any numerical value.
    - Like ordinal variables, but their categories cannot be ordered.
    - Example:
        - Name; Hair colour; Gender; Country; etc.

Graphical Displays for a Single Variable

- Continuous variable:
  - ***Histogram***:
    - An approximate representation of the distribution of numerical data.
    - Most used graph to show frequency distributions.
    - Example:

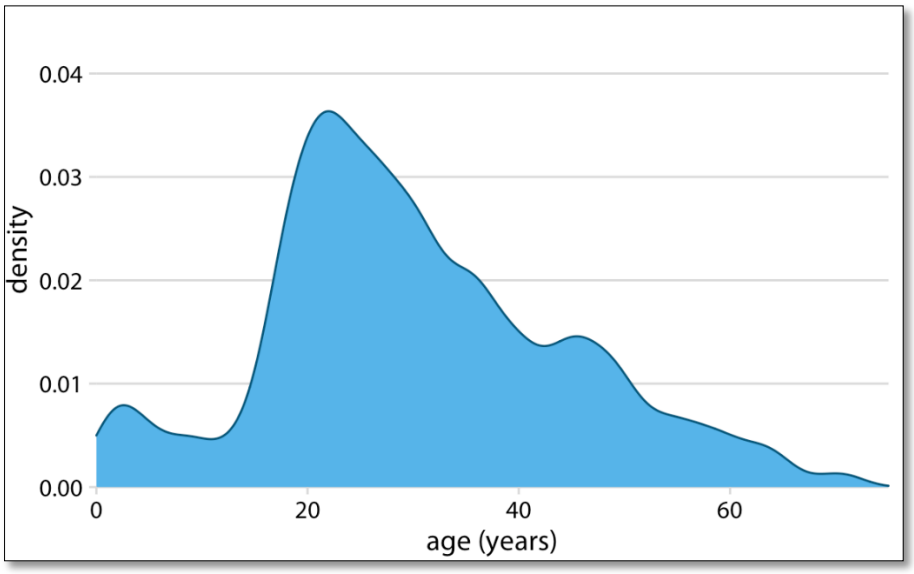


Histogram of the ages of Titanic passengers

[Source: Figure 7.1 taken from Wilke (2017)]

  - ***Kernel Density Plot***:
    - A representation of the distribution of numeric variable.
    - It uses a kernel density estimate to show the probability density function of the variable.
    - It is a smoothed version of the histogram and is used in the same concept.

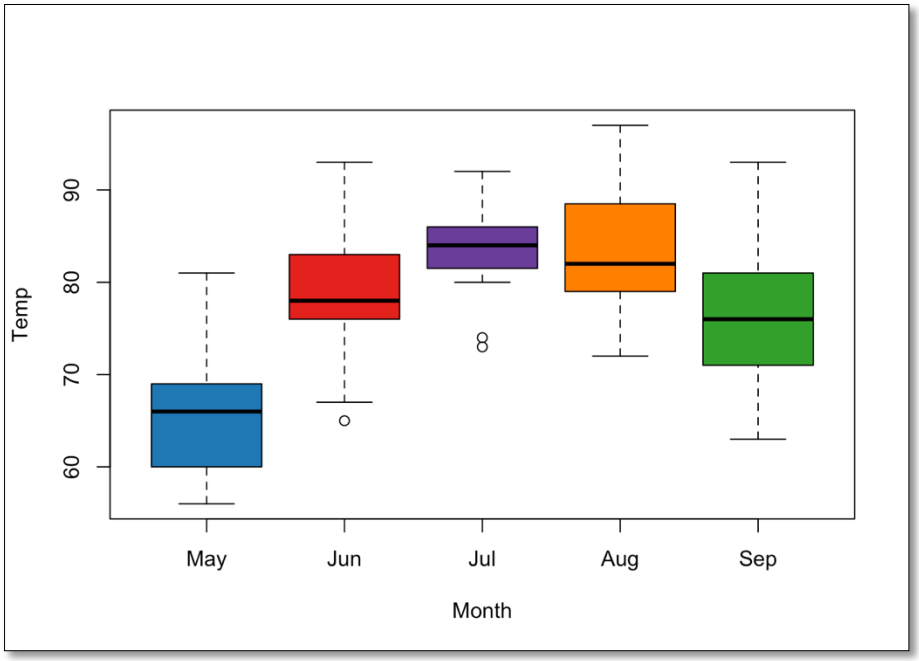
- Example:

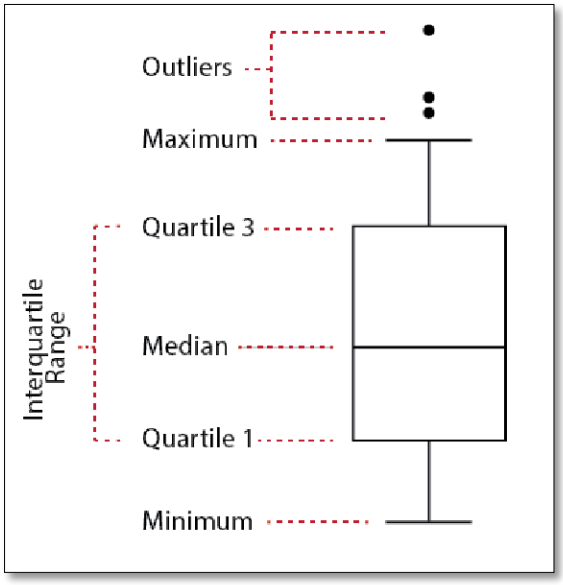


Kernel density plot of the ages of Titanic passengers

[Source: Figure 7.3 taken from Wilke (2017)]

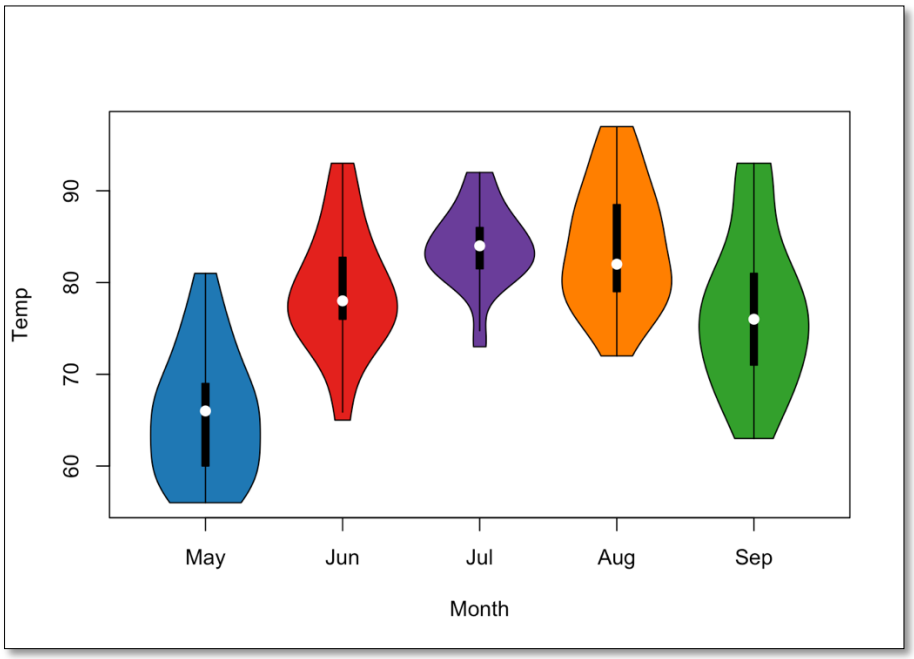
- ***Box Plot***:
  - A method for graphically demonstrating the locality, spread and skewness groups of numerical data through their quartiles.
  - Example:



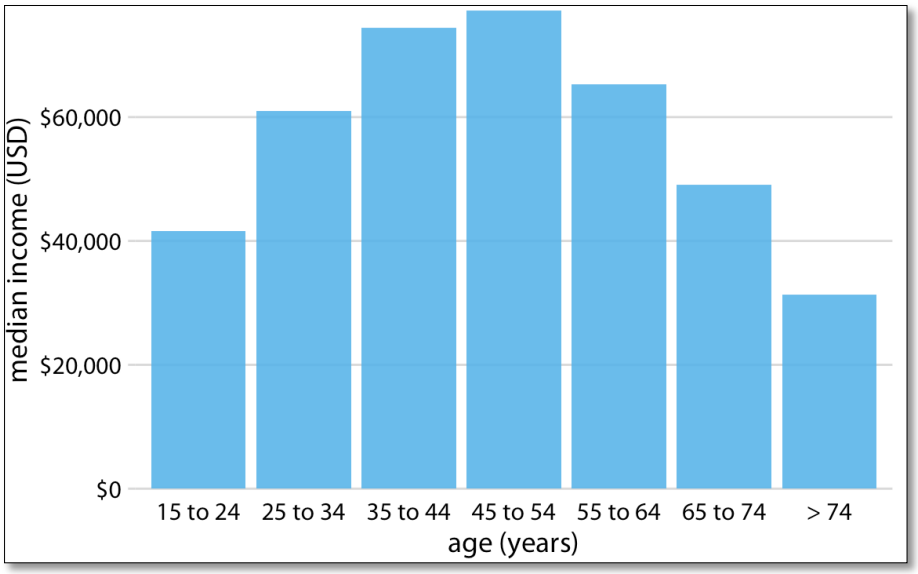


[Source: https://www.researchgate.net/figure/Boxplot-Legend-Diagram-explaining-the-statistical-definition-of-boxplots-referenced-in_fig2_344010487]

- ***Violin Plot***:
  - A method of plotting numeric data and can be considered a combination of the box plot with a rotated kernel density plot on each side.
  - Example:



- Categorical ordinal variable:
  - ***Bar Plot***:
    - Also referred to as ***bar chart*** or ***bar graph***.
    - Presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.
    - The bars can be plotted vertically or horizontally.
    - Can also be used for categorical nominal variable.
    - Example:



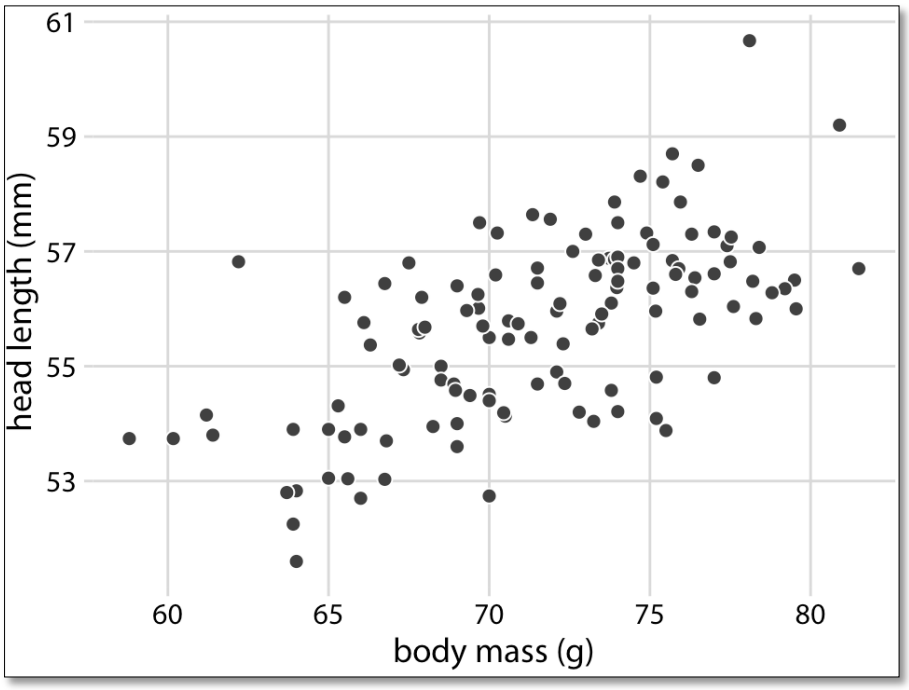
Bar plot on US annual household income vs. age group in 2016

[Source: Figure 6.5 taken from Wilke (2017)]

Graphical Displays for Two Variables

- Two continuous variable:
  - ***Scatter Plot***:
    - Using Cartesian coordinates (dots) to display values for two different numeric variables.
    - Primarily uses to observe and show relationships, the dots show the patterns when the data are taken as a whole.

- Example:

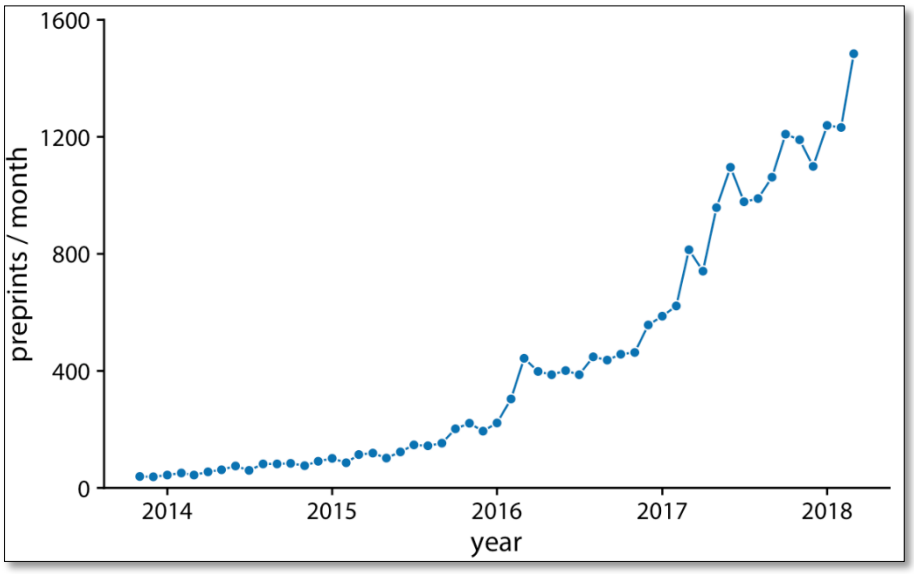


Head length vs. body mass

[Source: Figure 12.5 taken from Wilke (2017)]

- ***Line Plot***:
  - Also referred to as ***line chart*** or ***line graph***.
  - Displays information as a series of data points called markers connected by straight line segments.
  - Used to show information that changes over time.

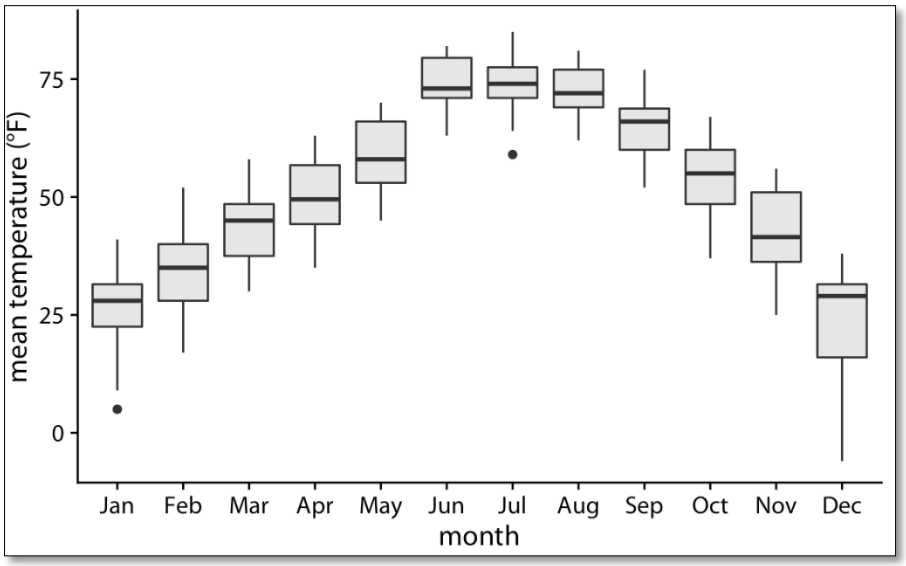
- Example:



Number of submissions per month to bioRxiv Nov 2014 and Apr 2018

[Source: Figure 13.2 taken from Wilke (2017)]

- ***Box Plot***:
  - Continuous variable *y*, categorical variable *x*.
  - Example:

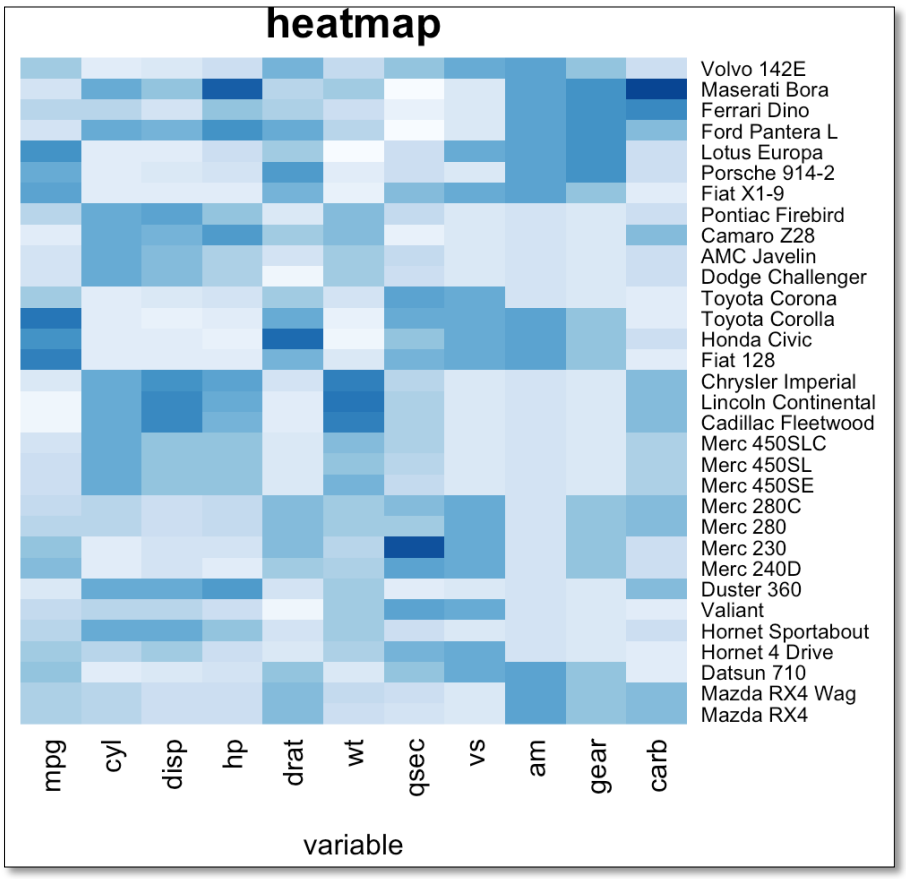


Mean daily temperatures in Lincoln, Nebraska, visualized as boxplots

[Source: Figure 9.3 taken from Wilke (2017)]

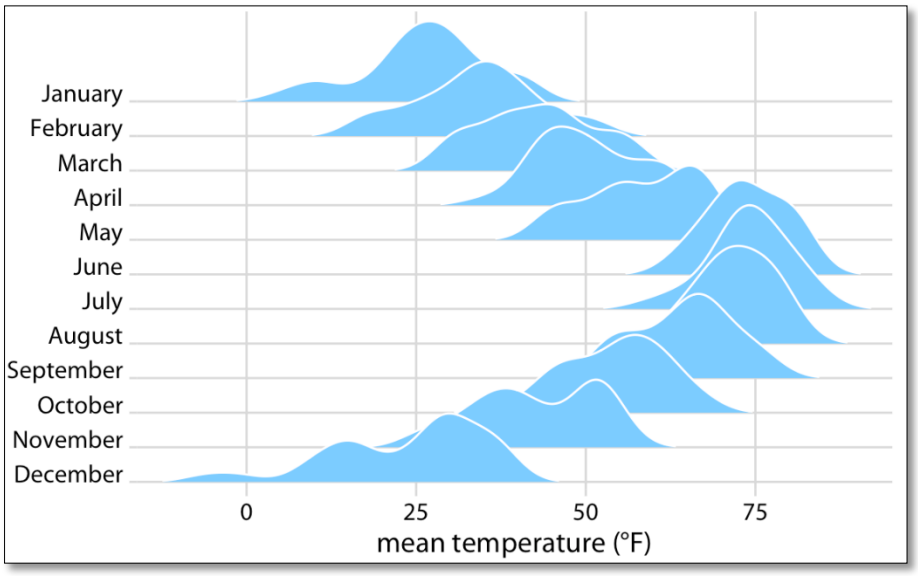
- ***Heat Map***:

- Shows magnitude of a phenomenon as colour in two dimensions.
- The variation in colour may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.
- Example:



- ***Ridgeline Plot***:
  - Also referred to as ***joy plot***.
  - Visualizes distributions of several groups of a category, each category or group of a category produces a density curve overlapping with each other.

- Example:



Mean daily temperatures in Lincoln, Nebraska, visualized as ridgeline plot

[Source: Figure 9.9 taken from Wilke (2017)]

Graphical Displays for Three or More Variables

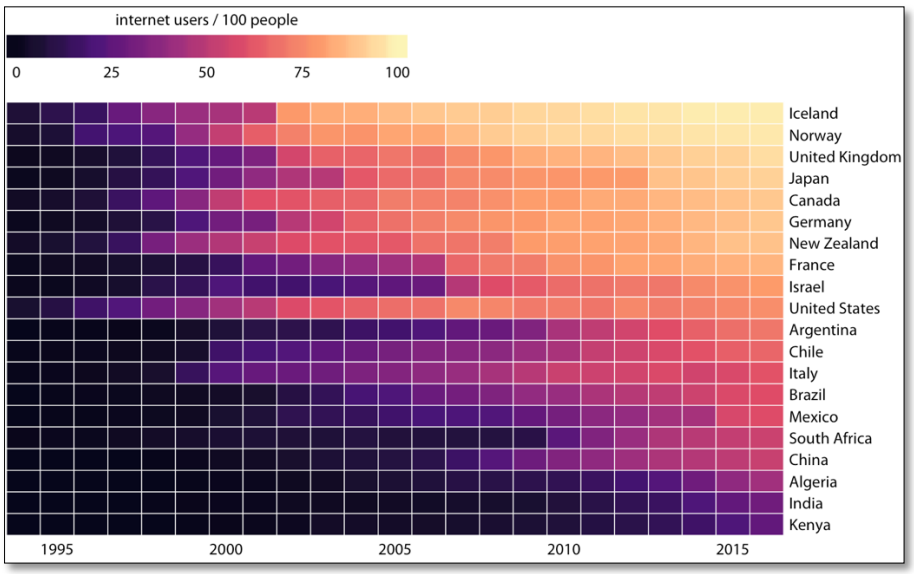
- Three or more continuous variable:
  - ***Matrix Scatter Plot***:
    - A grid of scatter plot used to visualize bivariate relationships between combinations of variables.
    - Each scatter plot in the matrix visualizes the relationship between a pair of variables, allowing many relationships to be explored in one chart.

- Example, 3 continuous 1 categorical variables:

Matrix scatter plot matrix of head length, body mass, and skull size, for 123 blue jays

[Source: Figure 12.4 taken from Wilke (2017)]

- ***Heat Map***:
  - Example, 2 categorical variables:

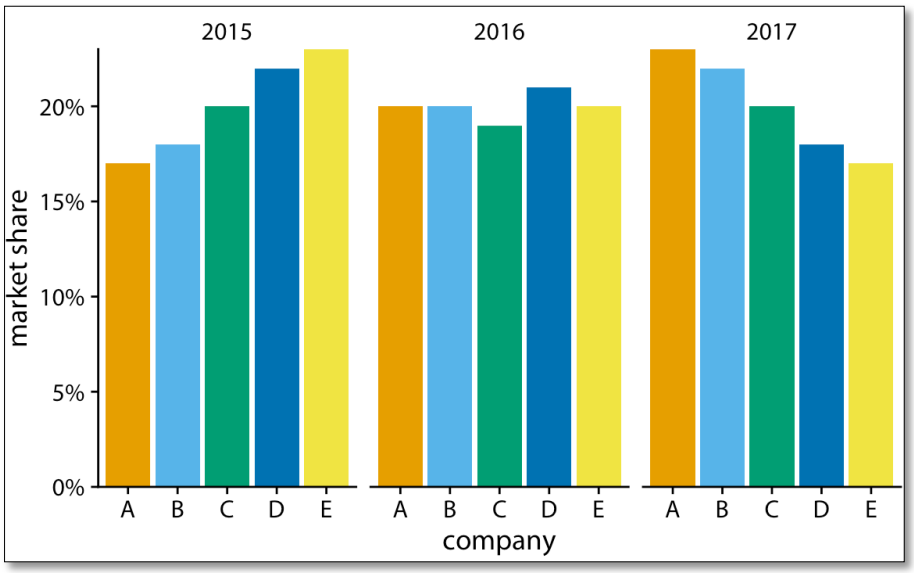


Internet use across countries and time; the colour reflects the

percentage of internet users for each country and year combination

[Source: Figure 6.14 taken from Wilke (2017)]

- ***Stacked Bar Plot***:
  - Extends the standard bar plot from looking at numeric values across one categorical variable to two.
  - Each bar in a standard bar plot is divided into a number of sub-bars stacked end to end, each one corresponding to a level of the second categorical variable.
- ***Group Bar Plot***:
  - ***Group Bar Plot*** also referred to as ***clustered bar plot*** or ***multi-series bar plot***.
  - Extends the bar chart, plotting numeric values for levels of two categorical variables instead of one.
  - Bars are grouped by position for levels of one categorical variable, with colour indicating the secondary category level within each group.
  - Example, 3 categorical variables:



Market share corresponding to 5 companies for the years 2015-2017

[Source: Figure 10.6 taken from Wilke (2017)]

2. Plotting in Python

- ***matplotlib***: basic yet comprehensive plotting library.
- ***seaborn***: prettier, high level interface data visualisation library based on matplotlib.
    - Import the libraries and reset the colour palette to colour-blind friendly

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

sns.reset_orig()
my_palette = sns.color_palette("colorblind")
plt.style.use("seaborn-colorblind")
```

Import and Clean the Data

- GDP per capita data from *DBnomics*, "*gdp.csv*" and "*gdp_wide.csv*".

gdp.csv

```
year,value,country
2000-01-01,27130.0,UK
2001-01-01,27770.0,UK
2002-01-01,28250.0,UK
2003-01-01,29060.0,UK
2004-01-01,29560.0,UK
2005-01-01,30210.0,UK
2006-01-01,30810.0,UK
2007-01-01,31280.0,UK
2008-01-01,30940.0,UK
2009-01-01,29460.0,UK
```

gdp_wide.csv

```
country,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,
DE,28910.0,29370.0,29290.0,29100.0,29470.0,29730.0,30930.0,31920.0,32320.0,30580.0,31940.0,33
FR,28930.0,29290.0,29410.0,29440.0,30050.0,30320.0,30850.0,31400.0,31310.0,30250.0,30690.0,31
IT,27430.0,27950.0,27960.0,27850.0,28030.0,28090.0,28480.0,28730.0,28230.0,26590.0,26930.0,27
UK,27130.0,27770.0,28250.0,29060.0,29560.0,30210.0,30810.0,31280.0,30940.0,29460.0,29830.0,29
```

- Wine chemistry composition data from *scikit-learn*, "*wine.csv*".

wine.csv

```
alcohol,malic_acid,ash,alcalinity_of_ash,magnesium,total_phenols,flavanoids,nonflavanoid_phenols
14.23,1.71,2.43,15.6,127.0,2.8,3.06,0.28,2.29,5.64,1.04,3.92,1065.0,class_0
13.2,1.78,2.14,11.2,100.0,2.65,2.76,0.26,1.28,4.38,1.05,3.4,1050.0,class_0
13.16,2.36,2.67,18.6,101.0,2.8,3.24,0.3,2.81,5.68,1.03,3.17,1185.0,class_0
14.37,1.95,2.5,16.8,113.0,3.85,3.49,0.24,2.18,7.8,0.86,3.45,1480.0,class_0
13.24,2.59,2.87,21.0,118.0,2.8,2.69,0.39,1.82,4.32,1.04,2.93,735.0,class_0
14.2,1.76,2.45,15.2,112.0,3.27,3.39,0.34,1.97,6.75,1.05,2.85,1450.0,class_0
14.39,1.87,2.45,14.6,96.0,2.5,2.52,0.3,1.98,5.25,1.02,3.58,1290.0,class_0
14.06,2.15,2.61,17.6,121.0,2.6,2.51,0.31,1.25,5.05,1.06,3.58,1295.0,class_0
14.83,1.64,2.17,14.0,97.0,2.8,2.98,0.29,1.98,5.2,1.08,2.85,1045.0,class_0
13.86,1.35,2.27,16.0,98.0,2.98,3.15,0.22,1.85,7.22,1.01,3.55,1045.0,class_0
```

- Reading data into data frames.

```
import numpy as np
import pandas as pd
from datetime import datetime

gdp = pd.read_csv("gdp.csv")

# convert the year column from string to datetime object
gdp['year'] = gdp['year'].apply(lambda x : datetime.strptime(x,'%Y-%M-%d'))

# get the GDP per capita data for each country, reset index for each series
gdp_uk = gdp[gdp['country']=='UK'].reset_index()
gdp_fr = gdp[gdp['country']=='FR'].reset_index()
gdp_it = gdp[gdp['country']=='IT'].reset_index()
gdp_de = gdp[gdp['country']=='DE'].reset_index()

gdp_wide = pd.read_csv("gdp_wide.csv")
gdp_wide.set_index('country',inplace=True)      # set row labels as countries
gdp_wide.columns = gdp_wide.columns.astype(int) # set years with type int

wine = pd.read_csv("wine.csv")
```

  - The first part of "*gdp_uk*" data frame.

```
gdp_uk.head()
```

	index	year	value	country
0	0	2000-01-01 00:01:00	27130.0	UK
1	1	2001-01-01 00:01:00	27770.0	UK
2	2	2002-01-01 00:01:00	28250.0	UK
3	3	2003-01-01 00:01:00	29060.0	UK
4	4	2004-01-01 00:01:00	29560.0	UK

- The first part of "*gdp_wide*" data frame.

```
gdp_wide.head()
```

```
            2000     2001     2002     2003  ...     2016     2017     2018     2019
country                                      ...
DE       28910.0  29370.0  29290.0  29100.0  ...  34610.0  35380.0  35720.0  35840.0
FR       28930.0  29290.0  29410.0  29440.0  ...  31770.0  32380.0  32860.0  33270.0
IT       27430.0  27950.0  27960.0  27850.0  ...  26020.0  26490.0  26780.0  26920.0
UK       27130.0  27770.0  28250.0  29060.0  ...  32060.0  32430.0  32640.0  32870.0

[4 rows x 20 columns]
```

- The first part of "*wine*" data frame.

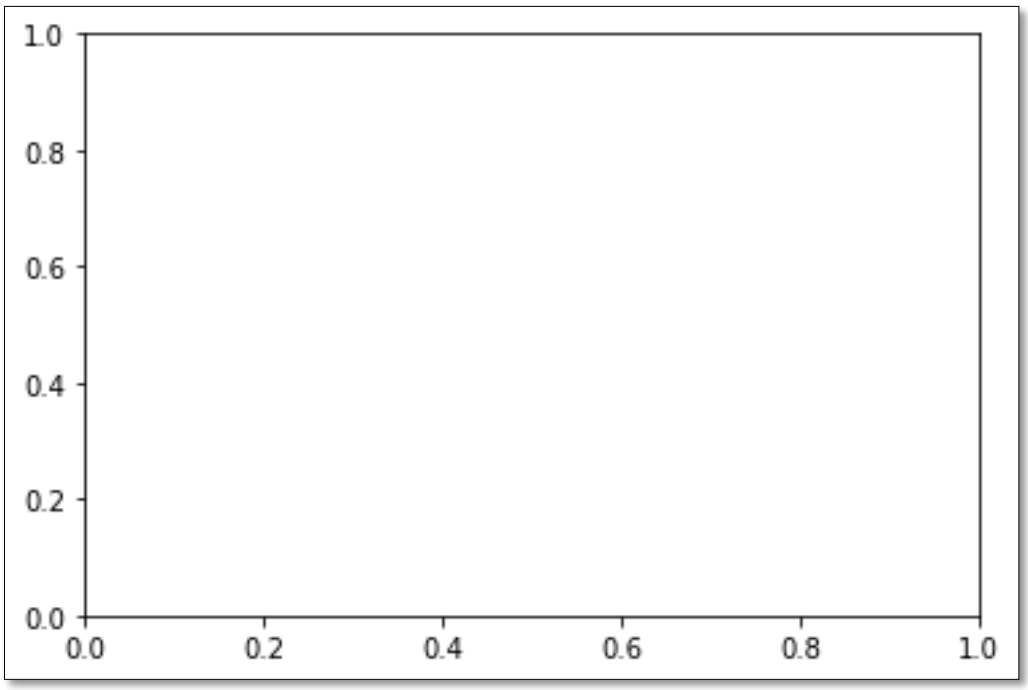
```
wine.head()
```

```
   alcohol  malic_acid   ash  ...  od280/od315_of_diluted_wines  proline   target
0    14.23        1.71  2.43  ...                          3.92   1065.0  class_0
1    13.20        1.78  2.14  ...                          3.40   1050.0  class_0
2    13.16        2.36  2.67  ...                          3.17   1185.0  class_0
3    14.37        1.95  2.50  ...                          3.45   1480.0  class_0
4    13.24        2.59  2.87  ...                          2.93    735.0  class_0

[5 rows x 14 columns]
```

Line Plot

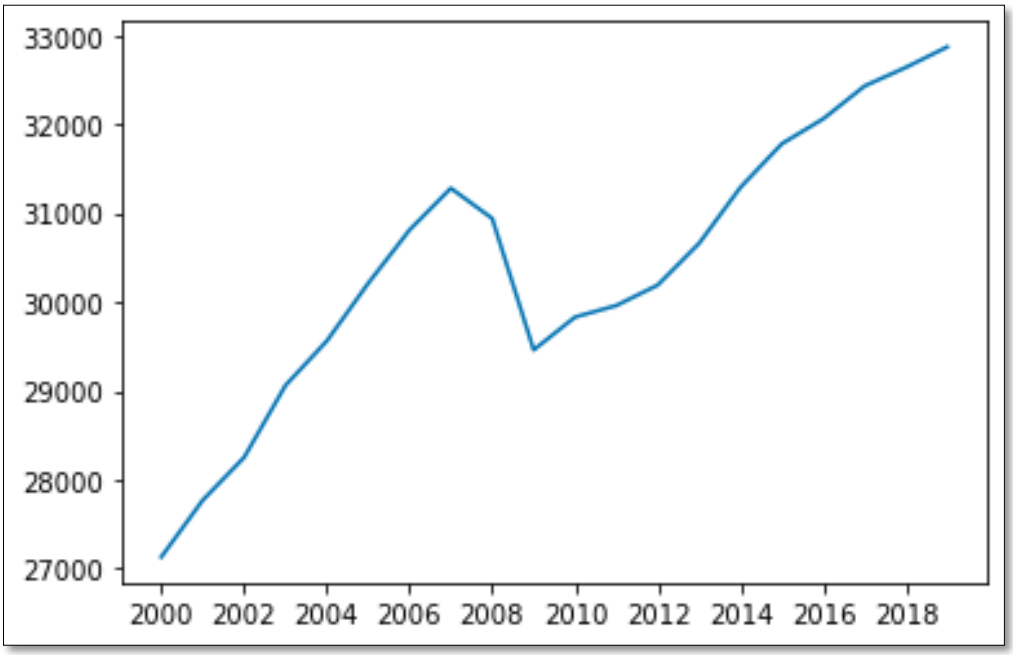
- Create an empty matplotlib plot, with empty figure and axes only.

```
fig, ax = plt.subplots()
```



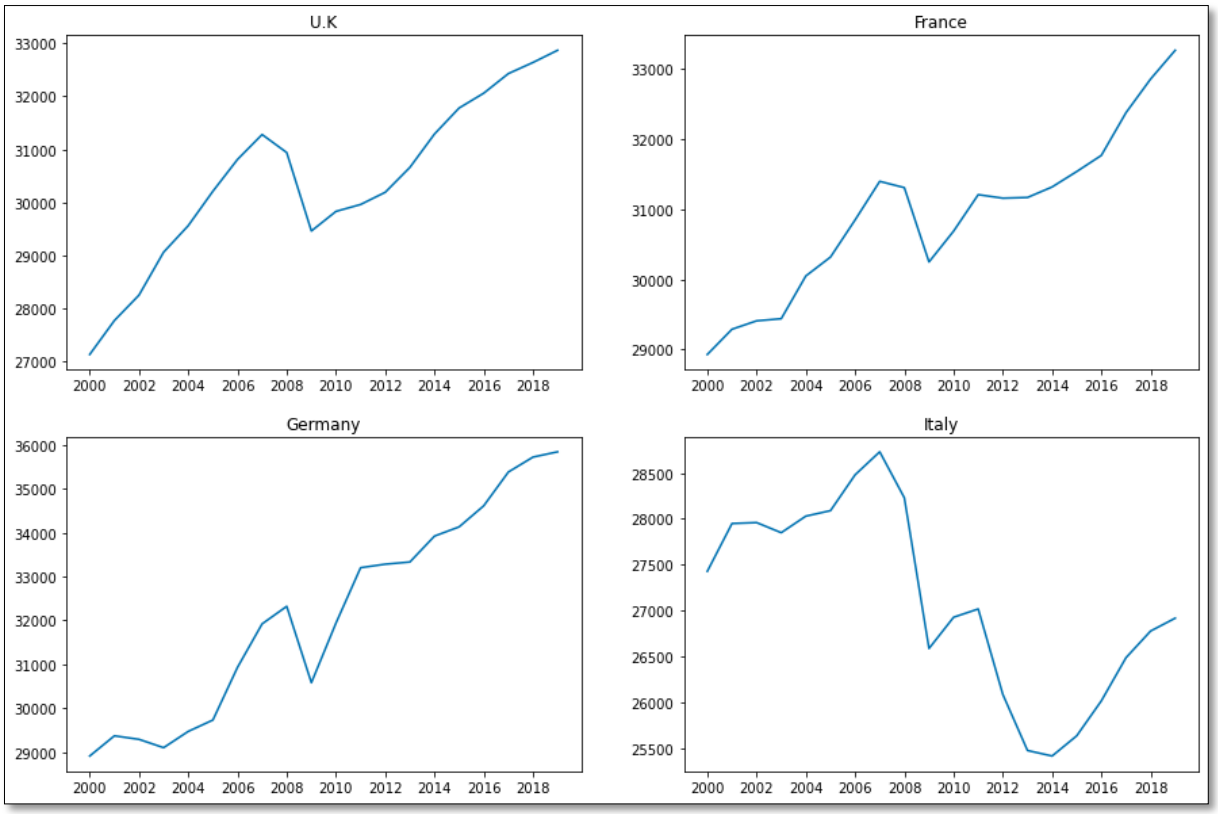
- Plot data using ***ax*** object and ***plot()*** method
  - x-axis: year, y-axis: value

```
fig, ax = plt.subplots()
ax.plot(gdp_uk['year'],gdp_uk['value'])
```



- Plotting multiple subplots

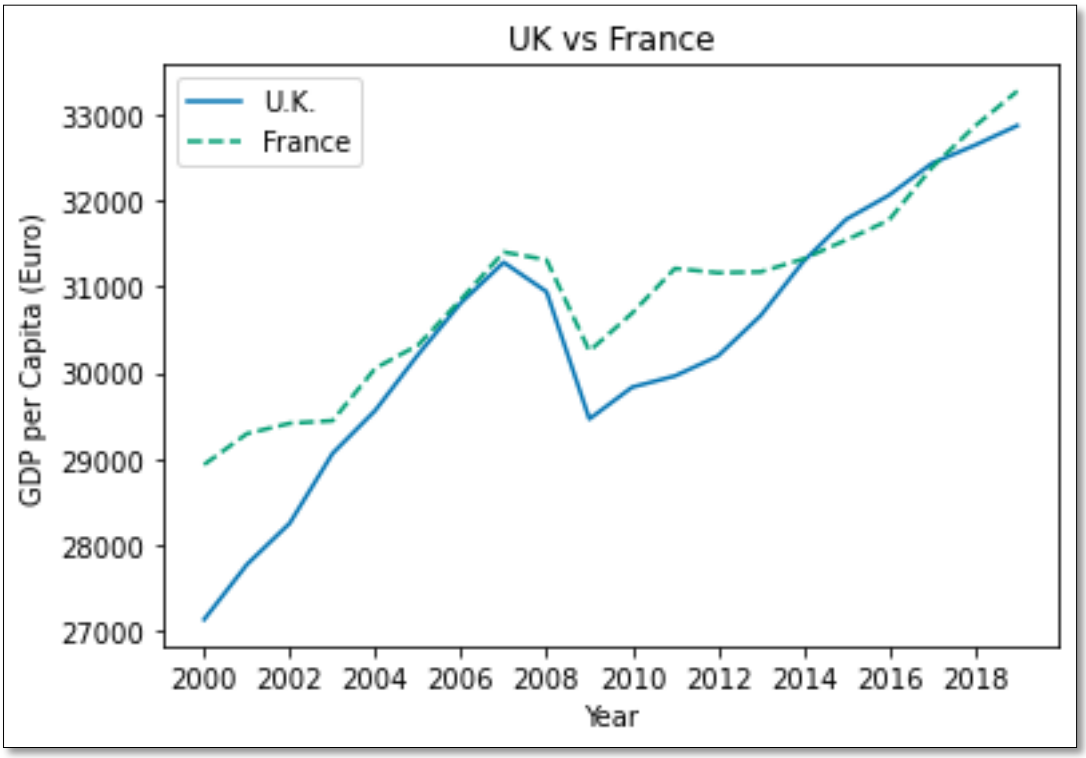
```
fig, ax = plt.subplots(2, 2, figsize = (15,10))      # ax is an array of an array (2x2)
                                                     # figsize 15in x 10in

ax[0][0].plot(gdp_uk['year'], gdp_uk['value'])       # top left
ax[0][0].title.set_text('U.K')

ax[0][1].plot(gdp_fr['year'], gdp_fr['value'])       # top right
ax[0][1].title.set_text('France')

ax[1][0].plot(gdp_de['year'], gdp_de['value'])       # bottom left
ax[1][0].title.set_text('Germany')

ax[1][1].plot(gdp_it['year'], gdp_it['value'])       # bottom right
ax[1][1].title.set_text('Italy')
```



- Plotting multiple lines on the same plot

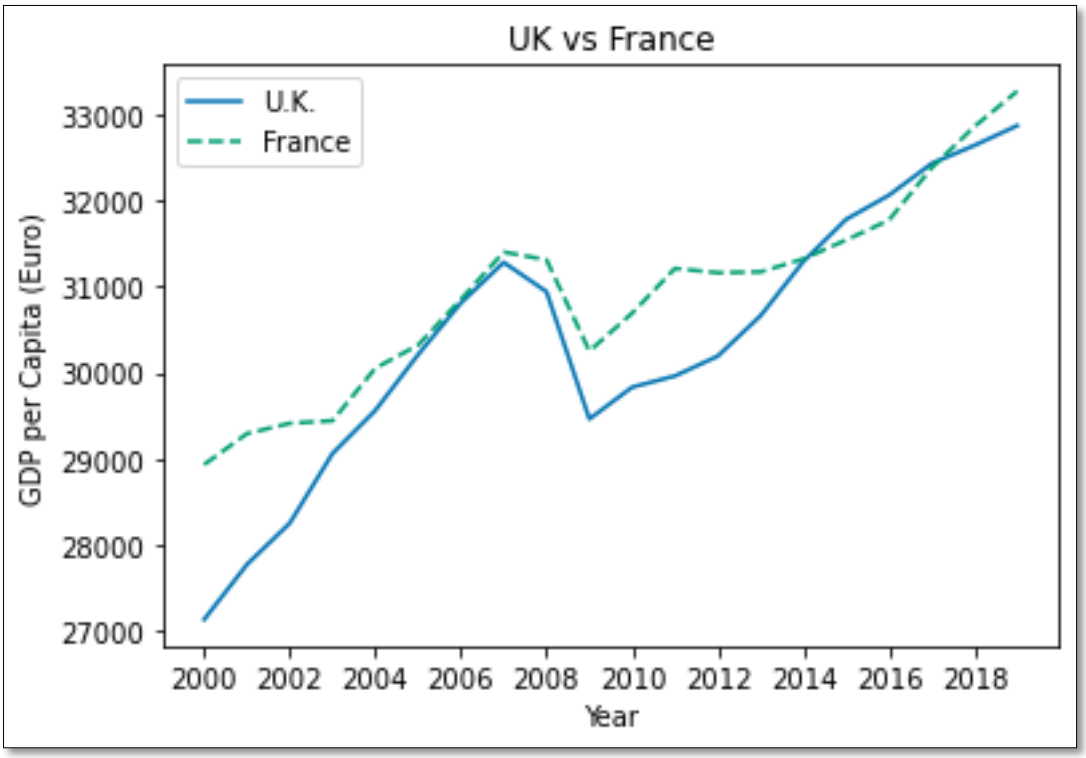
- With ***ax*** object ***plot()*** method:

```
fig, ax = plt.subplots()

# label will be shown on the legend, '--' for dashed line
ax.plot(gdp_uk['year'], gdp_uk['value'], label = 'U.K.')
ax.plot(gdp_fr['year'], gdp_fr['value'], '--', label = 'France')
ax.legend()

ax.set_xlabel("Year")
ax.set_ylabel("GDP per Capita (Euro)")
ax.title.set_text('UK vs France')
```



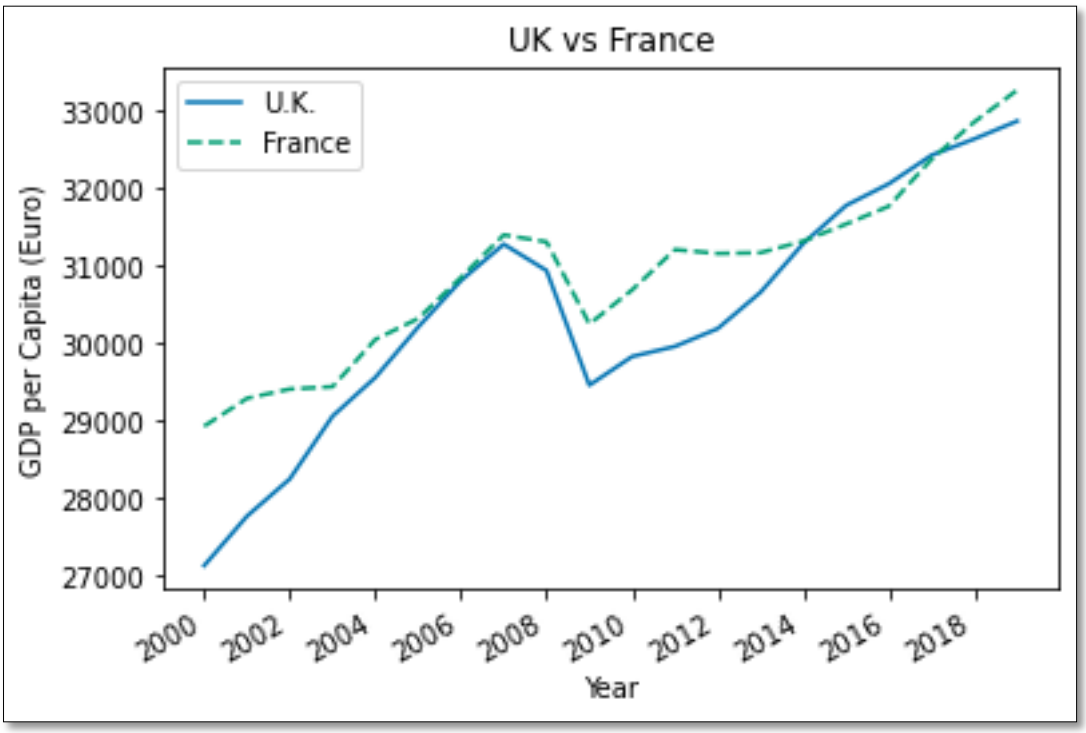
- With ***plt***:

```
plt.plot(gdp_uk['year'], gdp_uk['value'], label = 'U.K.')
plt.plot(gdp_fr['year'], gdp_fr['value'], '--', label = 'France')
plt.legend()

plt.xlabel('Year')
plt.ylabel('GDP per Capita (Euro)')
plt.title("UK vs France")
```



- With ***pandas*** data structure:

```
fig, ax = plt.subplots()

gdp_uk.plot(x = 'year', y = 'value', ax = ax)
gdp_fr.plot(x = 'year', y = 'value', style = '--', ax = ax)
ax.legend(["U.K.", "France"])

ax.set_xlabel('Year')
ax.set_ylabel('GDP per Capita (Euro)')
ax.title.set_text("UK vs France")
```

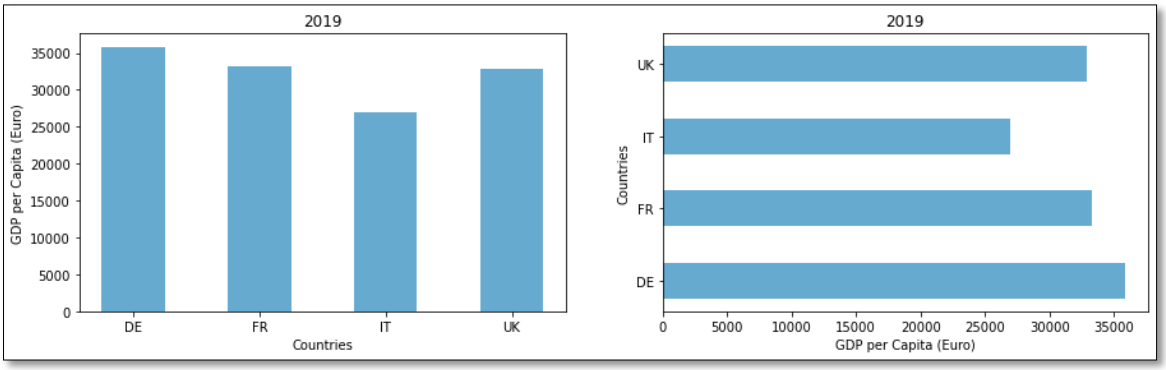


Bar Plot

- Vertical and horizontal bar plot:

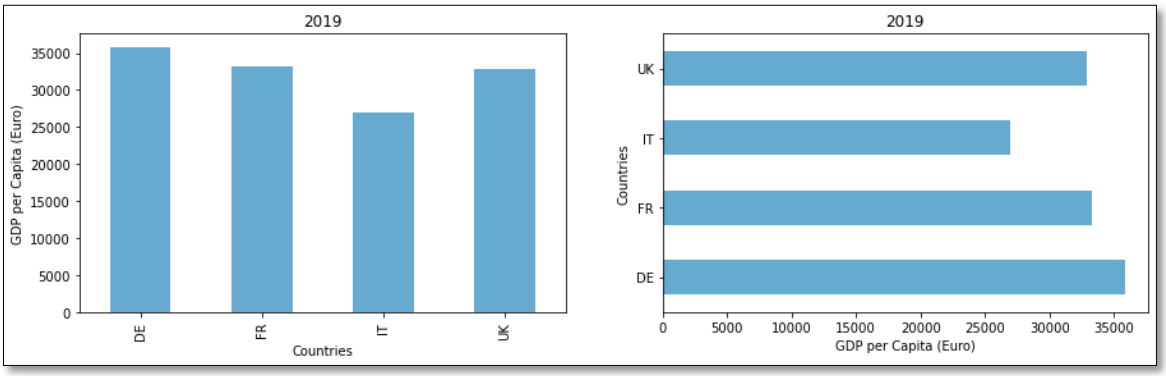
- With ***ax*** object ***bar()*** method:

```
fig, ax = plt.subplots(1, 2, figsize=(15, 4))

ax[0].bar(gdp_wide[2019].index,      # vertical bars
          gdp_wide[2019],
          alpha = 0.6,               # makes the bars slightly transparent
          width = 0.5)               # makes the bars thinner
ax[0].set_xlabel('Countries')
ax[0].set_ylabel('GDP per Capita (Euro)')
ax[0].title.set_text("2019")

ax[1].barh(gdp_wide[2019].index,     # horizontal bars
           gdp_wide[2019],
           alpha = 0.6,
           height = 0.5)             # 'height' to make the bars thinner
ax[1].set_xlabel('GDP per Capita (Euro)')
ax[1].set_ylabel('Countries')
ax[1].title.set_text("2019")
```



- With ***pandas***:

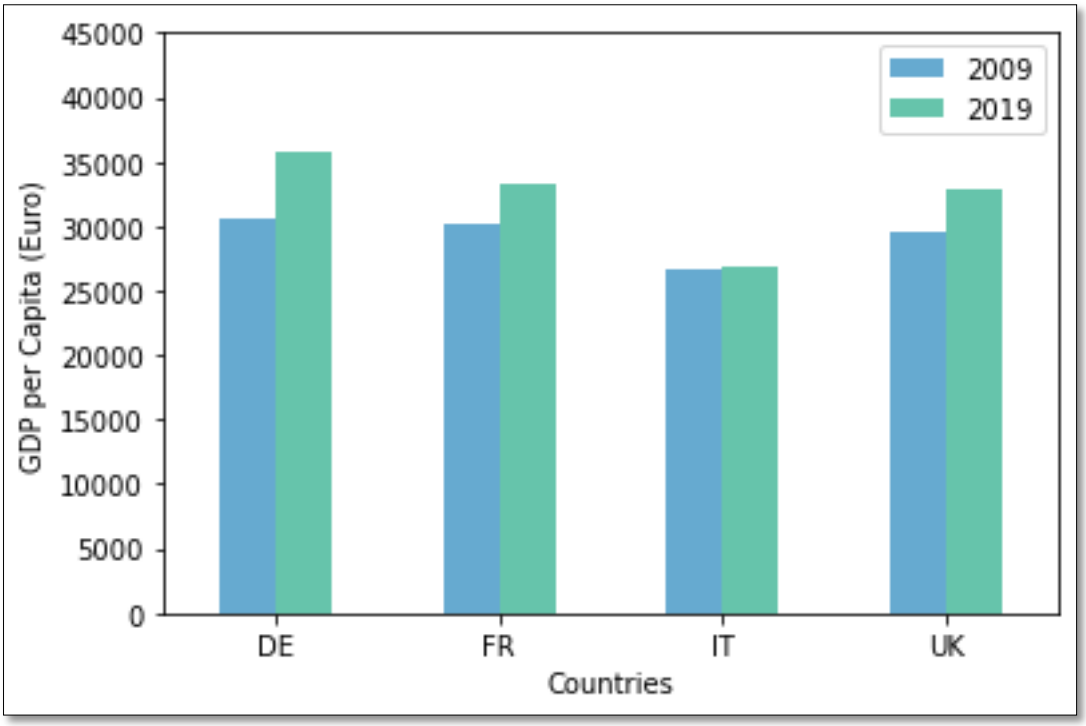
```
fig, ax = plt.subplots(1, 2, figsize=(15, 4))

gdp_wide[2019].plot.bar(x='country', y='value',      # vertical bars
                        ax = ax[0],
                        legend = False,              # no legend
                        alpha = 0.6)
ax[0].set_xlabel('Countries')
ax[0].set_ylabel('GDP per Capita (Euro)')
ax[0].title.set_text("2019")

gdp_wide[2019].plot.barh(x='country', y='value',     # horizontal bars
                         ax = ax[1],
                         legend = False,
                         alpha = 0.6)
ax[1].set_xlabel('GDP per Capita (Euro)')
ax[1].set_ylabel('Countries')
ax[1].title.set_text("2019")
```



- Side by side bar plot

- Using ***pandas*** data frame

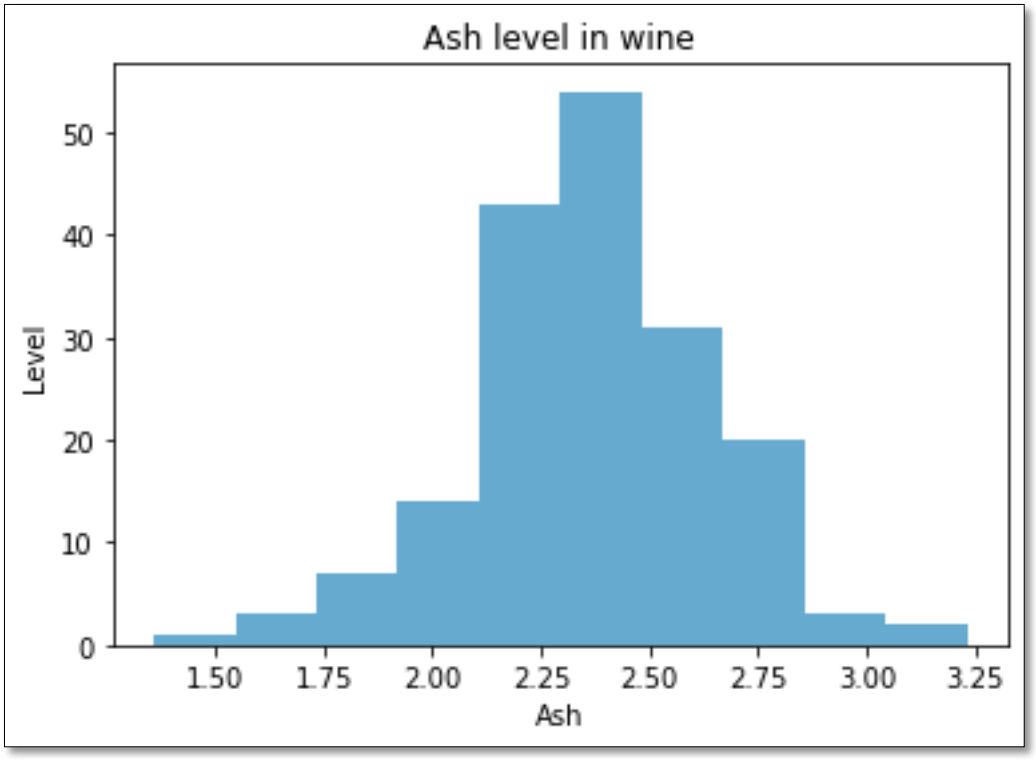
```
ax = gdp_wide[[2009,2019]].plot.bar(rot=0,      # rotate the label
                                    alpha=0.6)
ax.set_ylim(top = 45000)
ax.set_xlabel('Countries')
ax.set_ylabel('GDP per Capita (Euro)')
```



Histogram

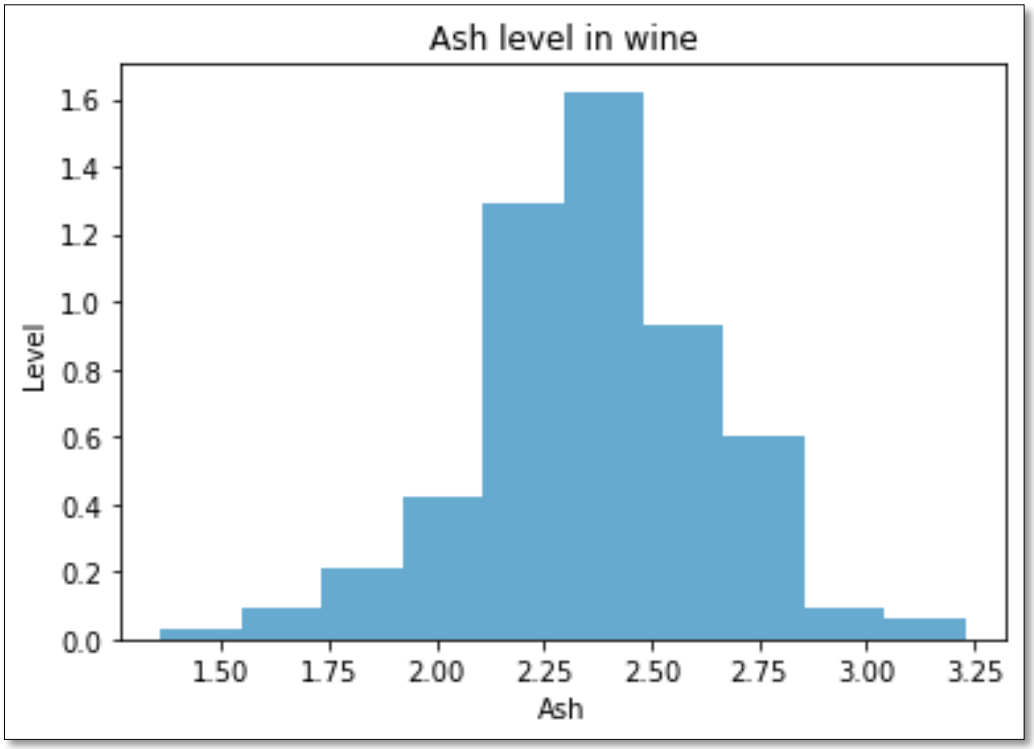
- Frequency distribution:

```
fig, ax = plt.subplots()

plt.hist(wine['ash'], 10, alpha = 0.6)  # bin size 10
plt.xlabel('Ash')
plt.ylabel('Level')
plt.title("Ash level in wine")
```


Ash level in wine
Level
0
10
20
30
40
50
1.50
1.75
2.00
2.25
2.50
2.75
3.00
3.25
Ash

- Density distribution:

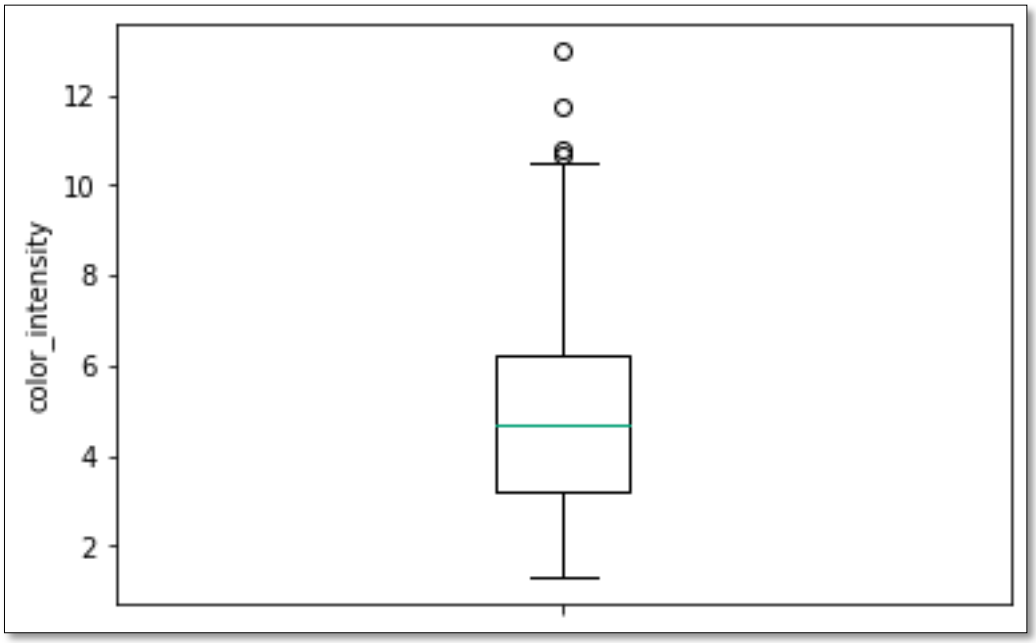
```
fig, ax = plt.subplots()

plt.hist(wine['ash'], 10, density = 1, alpha = 0.6)
plt.xlabel('Ash')
plt.ylabel('Level')
plt.title("Ash level in wine")
```



Boxplot

- Basic single boxplot:

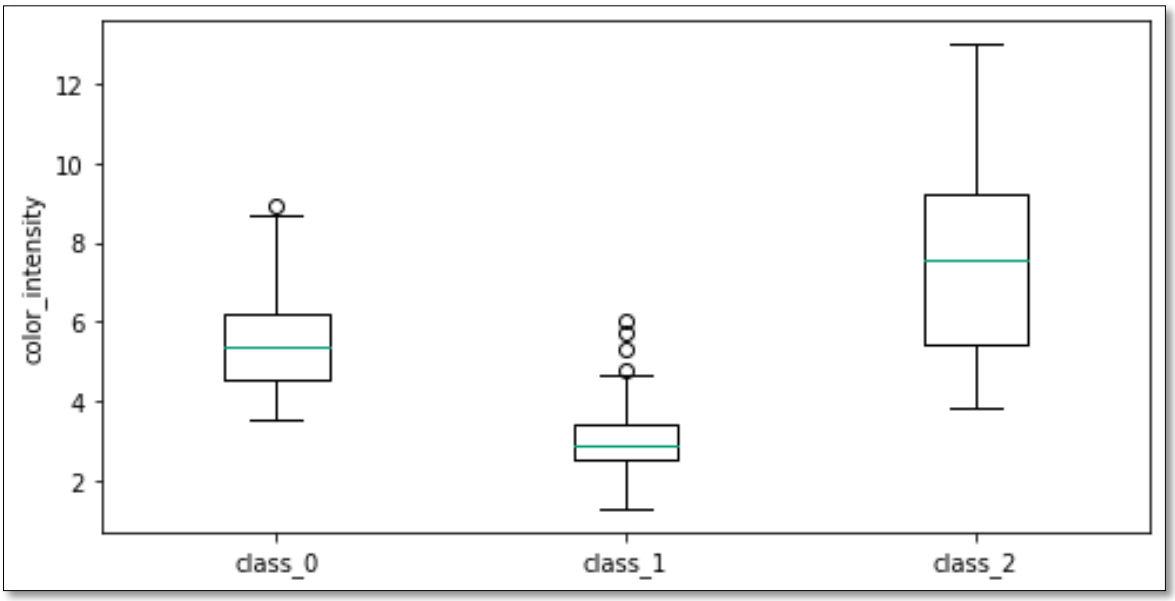
```
fig, ax = plt.subplots()

ax.boxplot(wine['color_intensity'])
ax.set_ylabel('color_intensity')
ax.set_xticklabels([''])    # no label for x-axis
```



- Side by side grouped boxplot

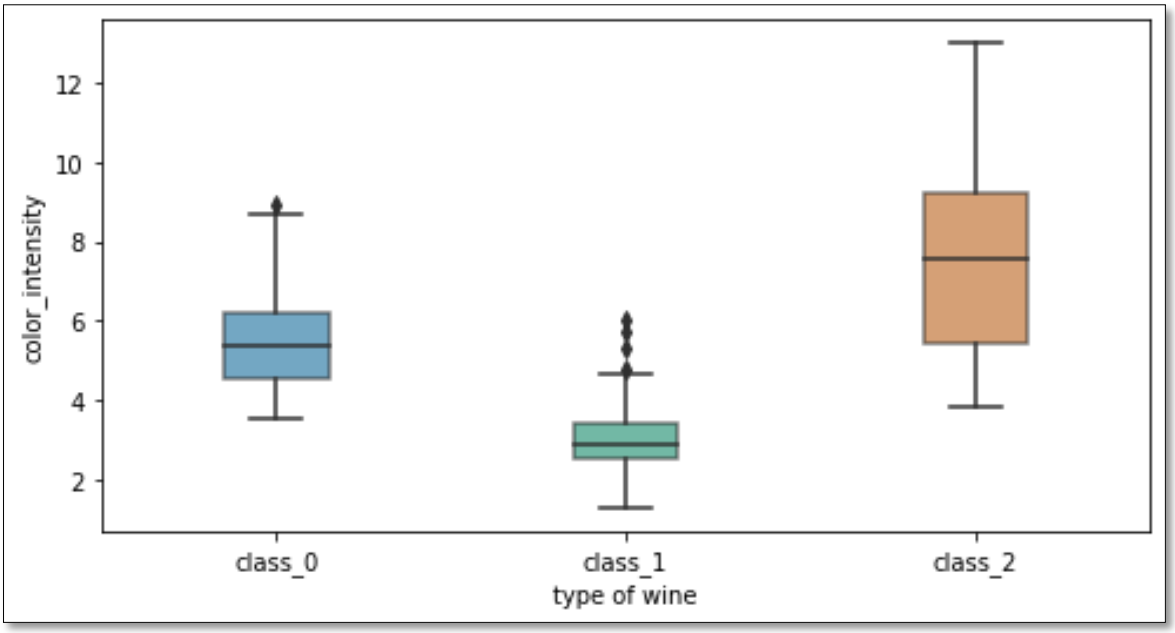
- Using ***matplotlib***:

```
fig, ax = plt.subplots(figsize = (8, 4))

ax.boxplot([wine['color_intensity'][wine['target']=='class_0'],
            wine['color_intensity'][wine['target']=='class_1'],
            wine['color_intensity'][wine['target']=='class_2']])
ax.set_ylabel('color_intensity')
ax.set_xticklabels(['class_0', 'class_1', 'class_2'])
```



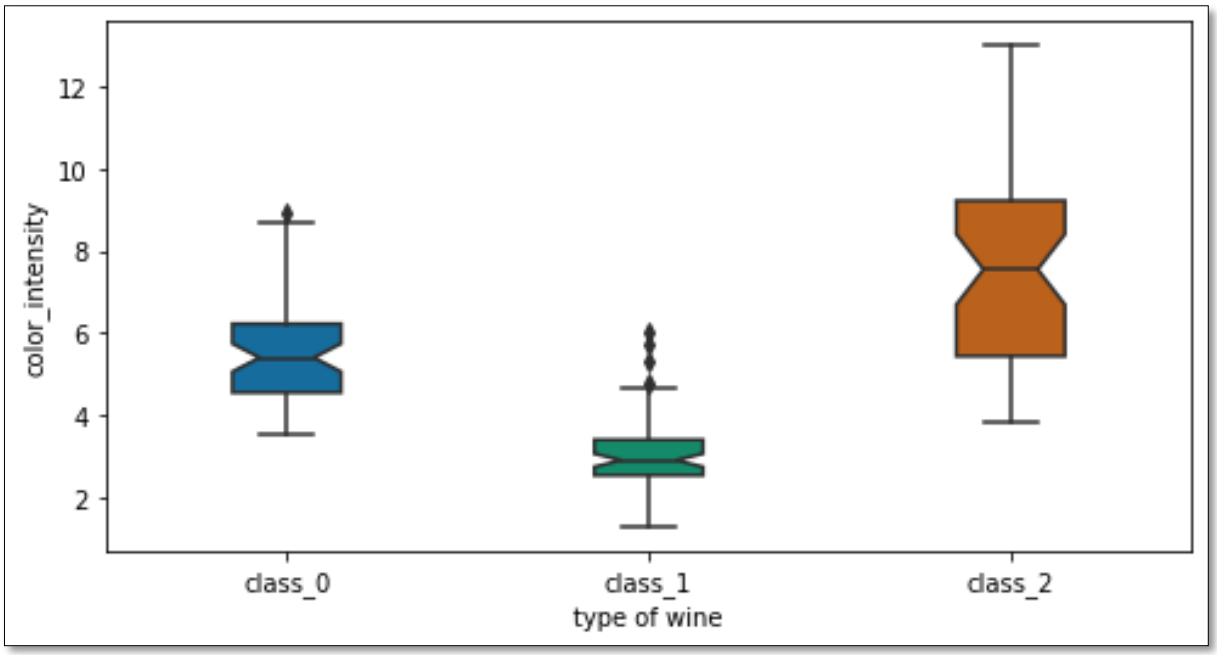
- Using ***seaborn***:

```
fig, ax = plt.subplots(figsize = (8, 4))

sns.boxplot(data = wine, x = 'target', y = 'color_intensity',
            width = 0.3, boxprops = dict(alpha=0.6))
ax.set_xlabel("type of wine")
```



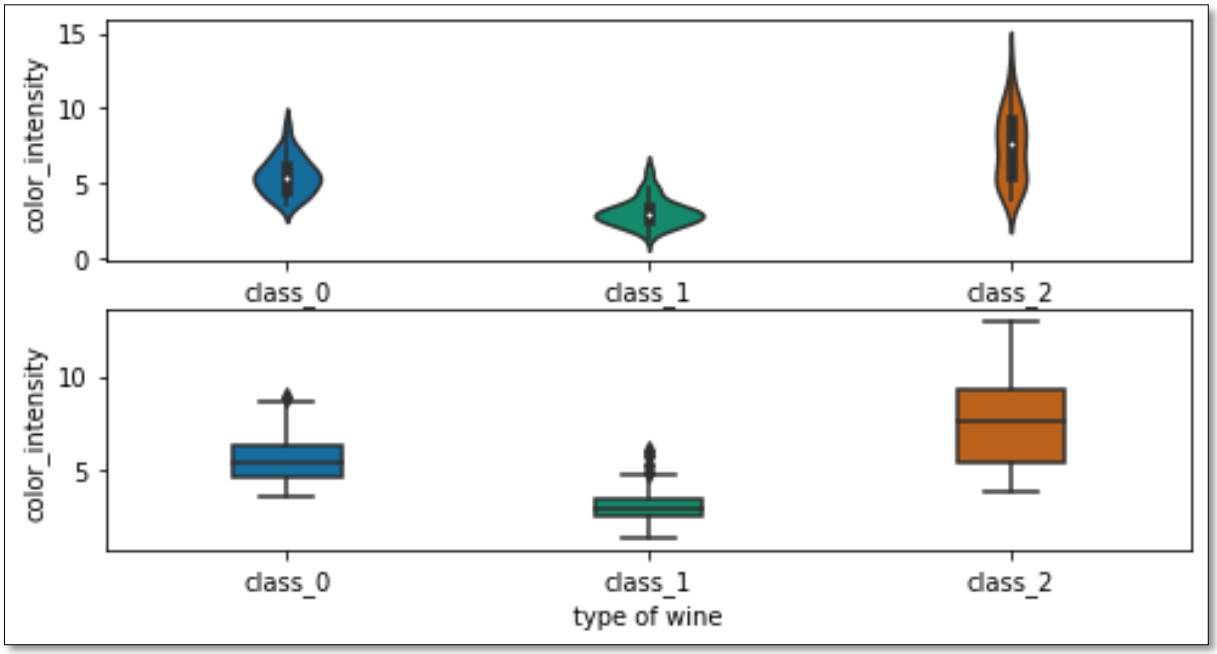
- Notch boxplot:

```
plt.subplots(figsize=(8, 4))
ax = sns.boxplot(data = wine, x = 'target', y = 'color_intensity',
                 notch = True, width = 0.3)
ax.set_xlabel("type of wine")
```



- Violin Plot:

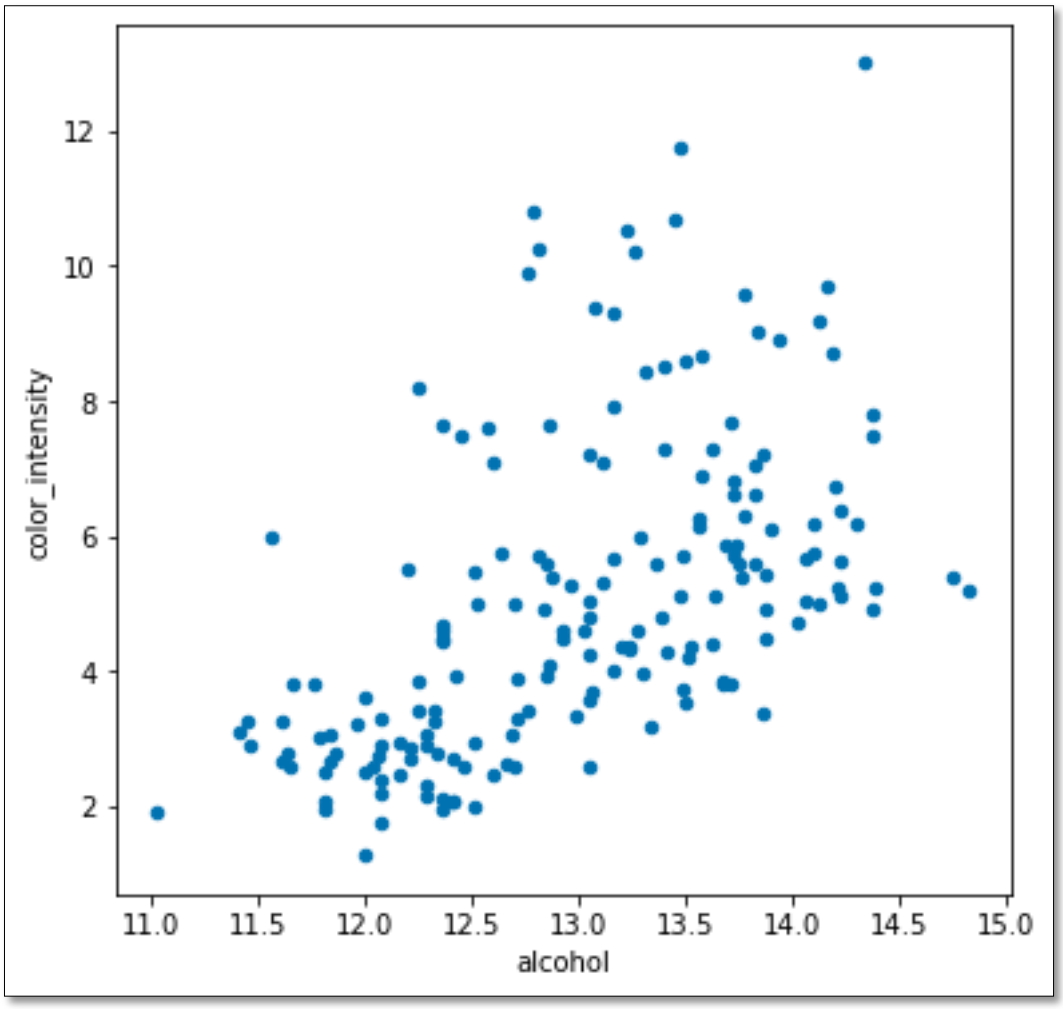
```
fig, ax = plt.subplots(2, 1, figsize=(8, 4))

sns.violinplot(data = wine, x = 'target', y = 'color_intensity',
               ax = ax[0], width = 0.3)
sns.boxplot(data = wine, x = 'target', y = 'color_intensity',
            ax = ax[1], width = 0.3)
ax[1].set_xlabel("type of wine")
```



Scatter Plot

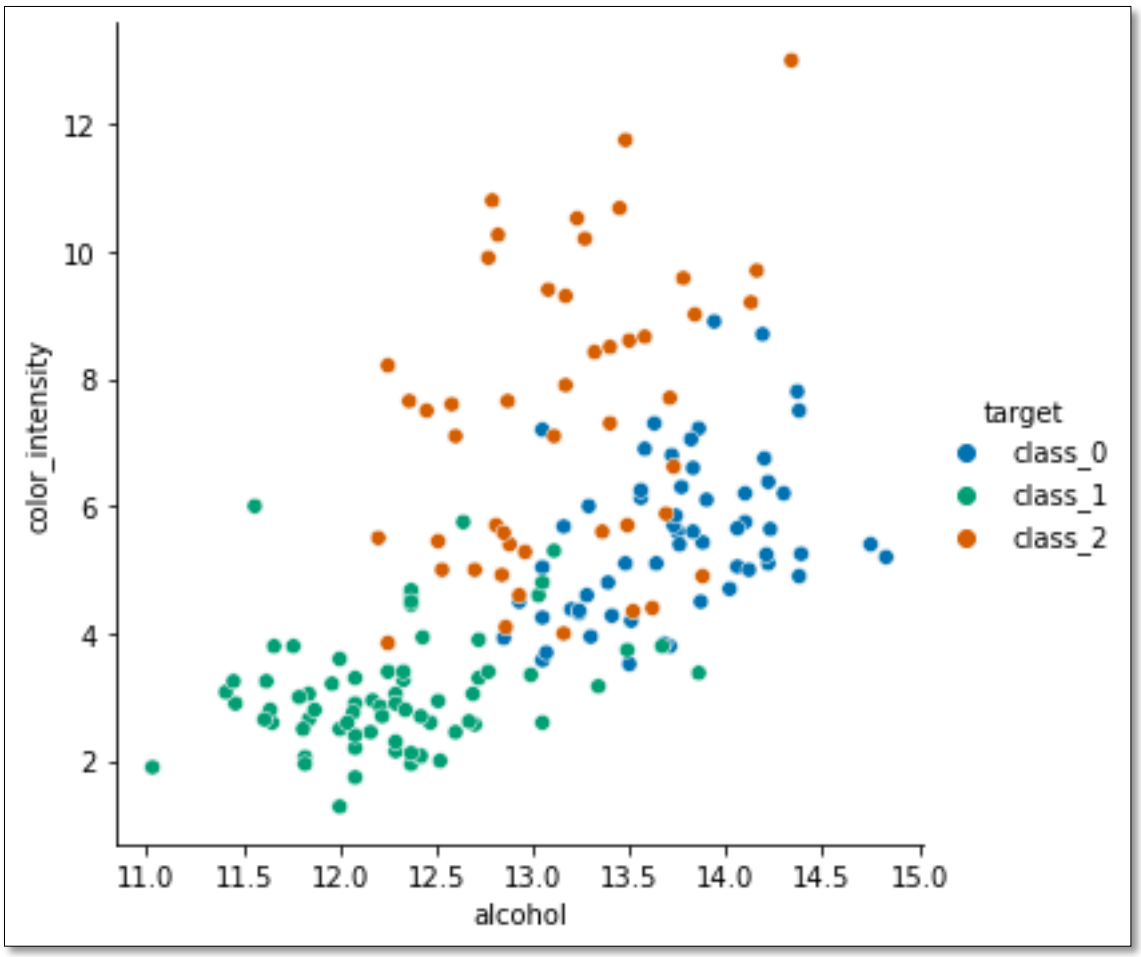
- Using ***matplotlib***:

```
ax = wine.plot.scatter('alcohol', 'color_intensity', figsize=(6, 6))
```

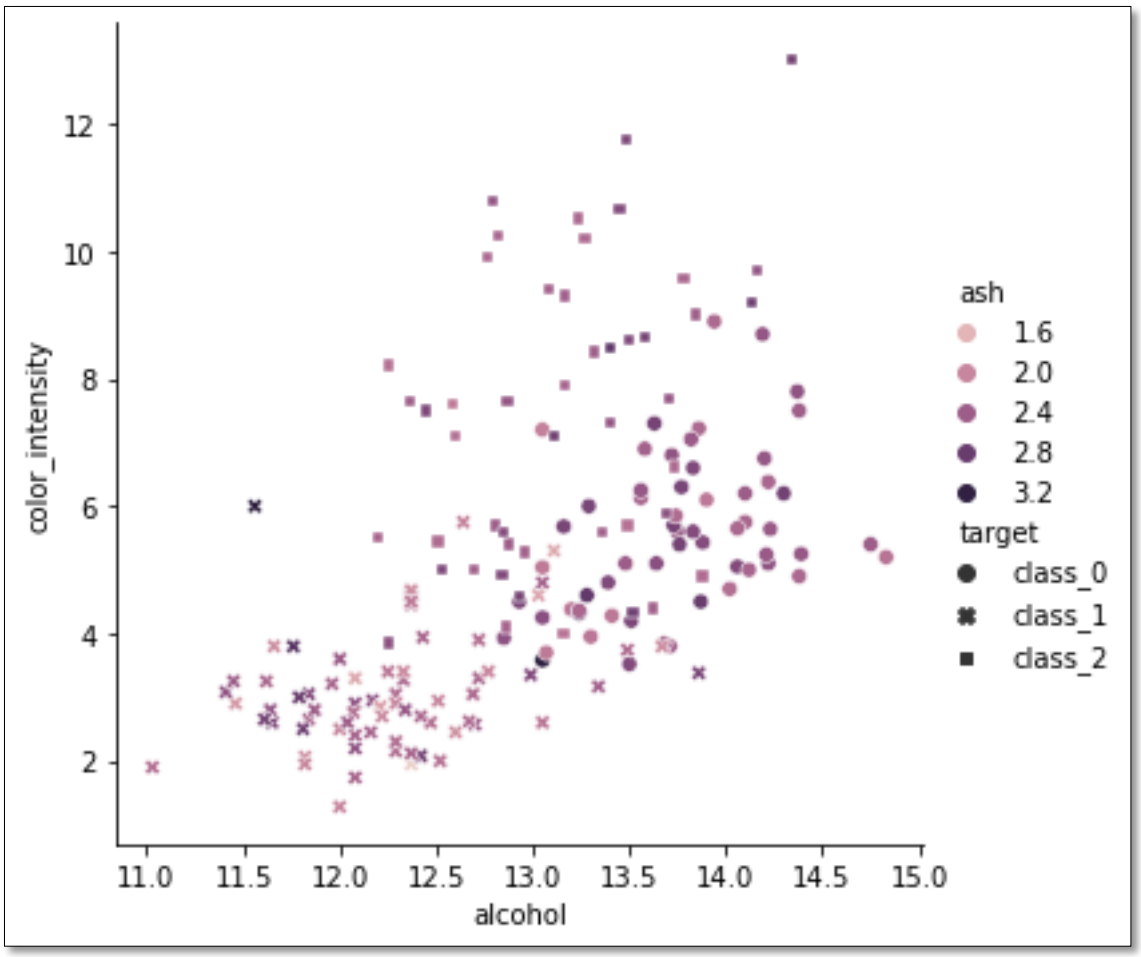


- Using ***seaborn***:

```
sns.relplot(x = 'alcohol', y = 'color_intensity', hue = "target", data = wine)
```

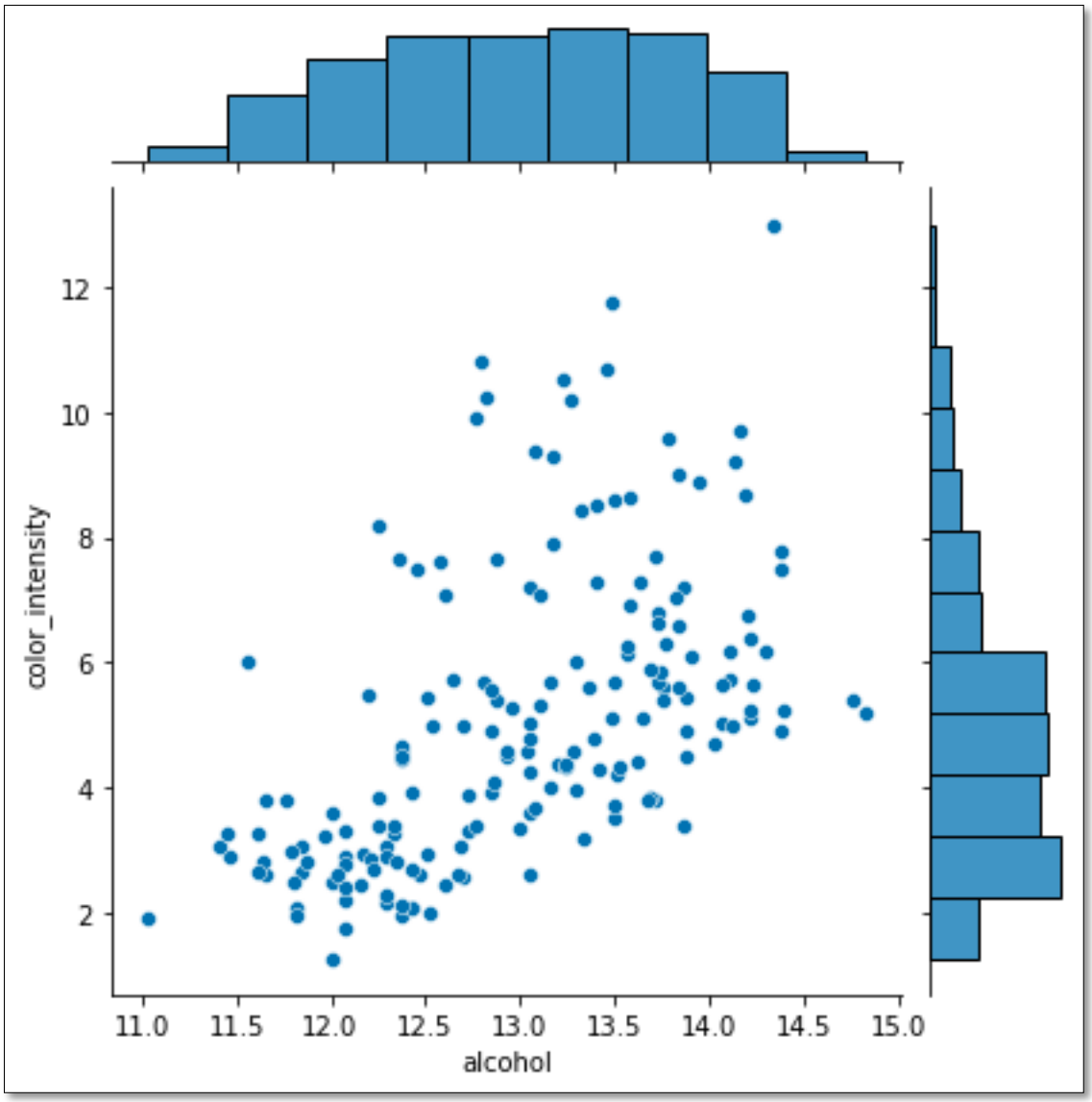



```
sns.relplot(x = 'alcohol', y = 'color_intensity',
            hue = "ash", style = 'target', data = wine)
```


color_intensity
alcohol
ash
1.6
2.0
2.4
2.8
3.2
target
class_0
class_1
class_2

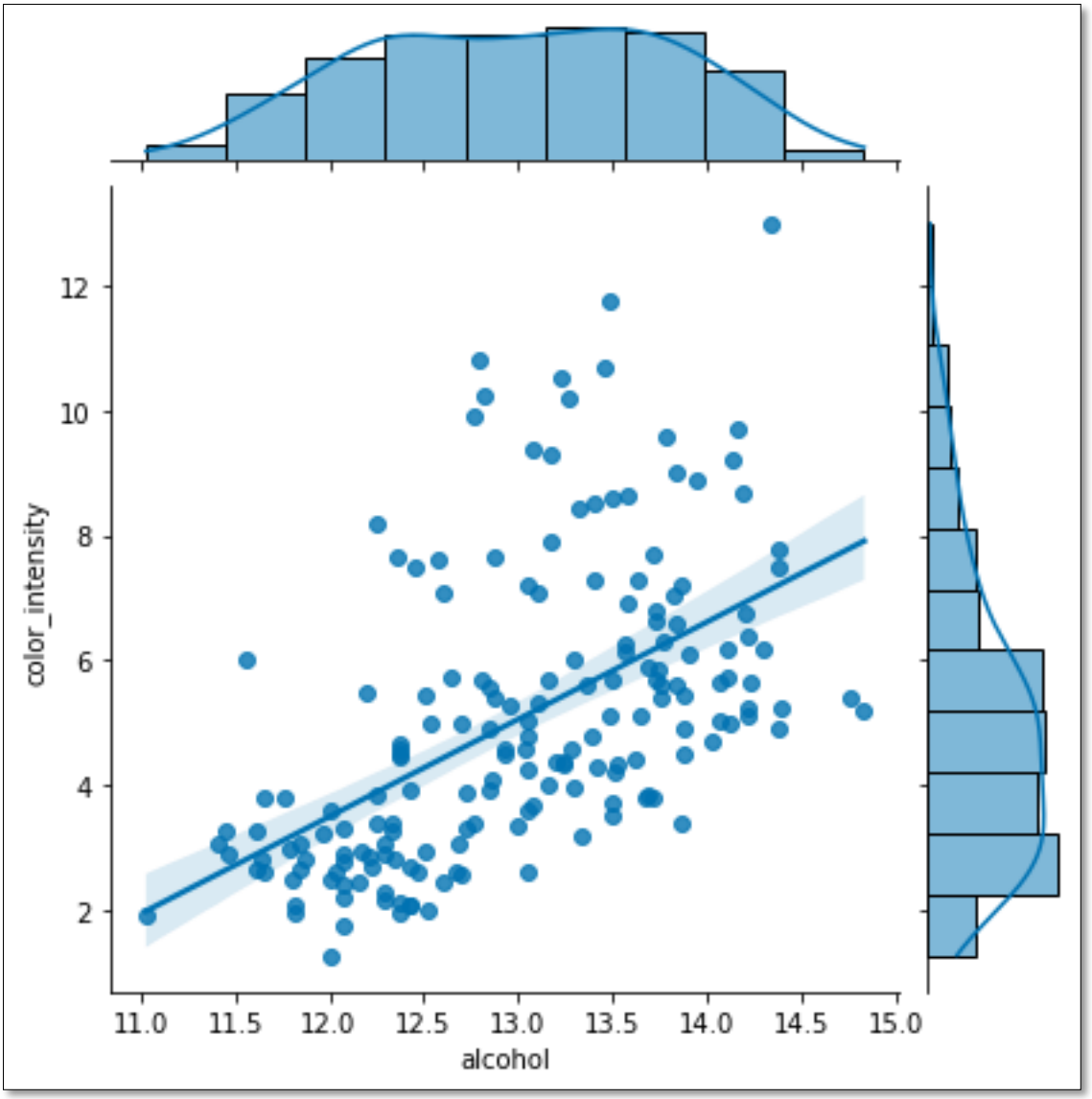
- Marginal histogram with scatter plot:

```
sns.jointplot(x = 'alcohol', y = 'color_intensity', data = wine)
```



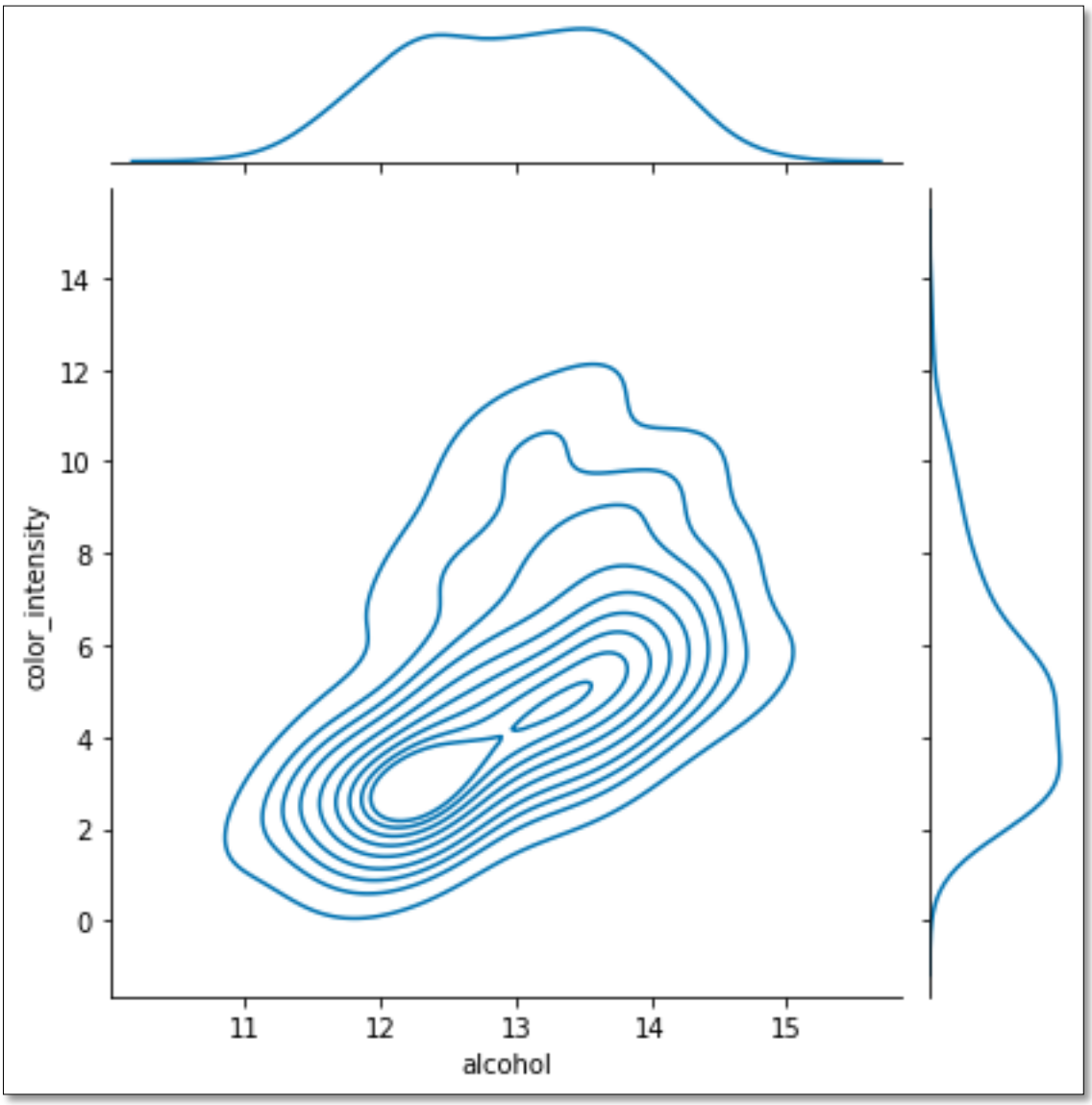
  - With regression:

```
sns.jointplot(x = 'alcohol', y = 'color_intensity', data = wine, kind = 'reg')
```


color_intensity
alcohol
12
10
8
6
4
2
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0

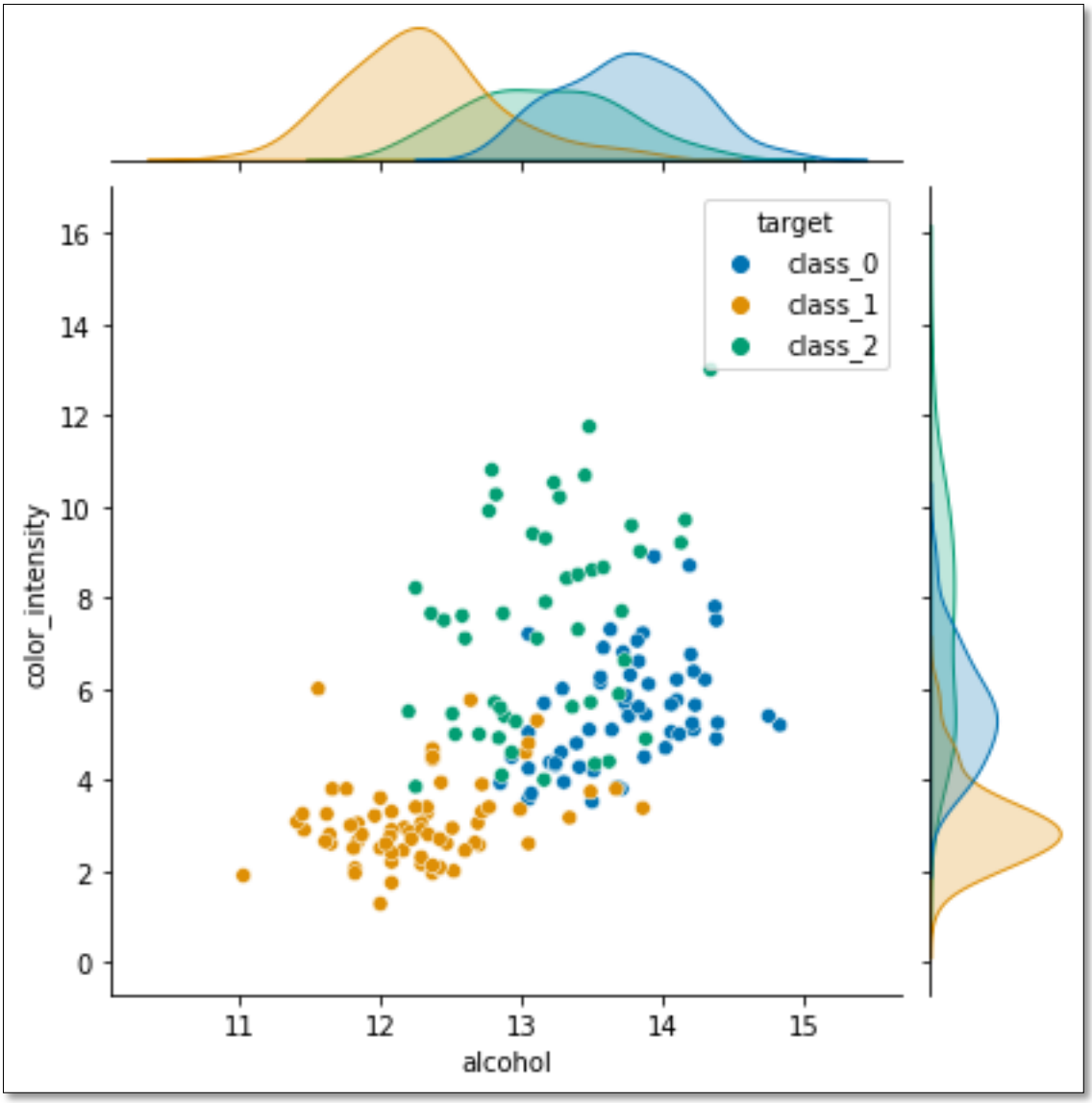
- With kernel density estimation:

```
sns.jointplot(x = 'alcohol', y = 'color_intensity', data = wine, kind = 'kde')
```

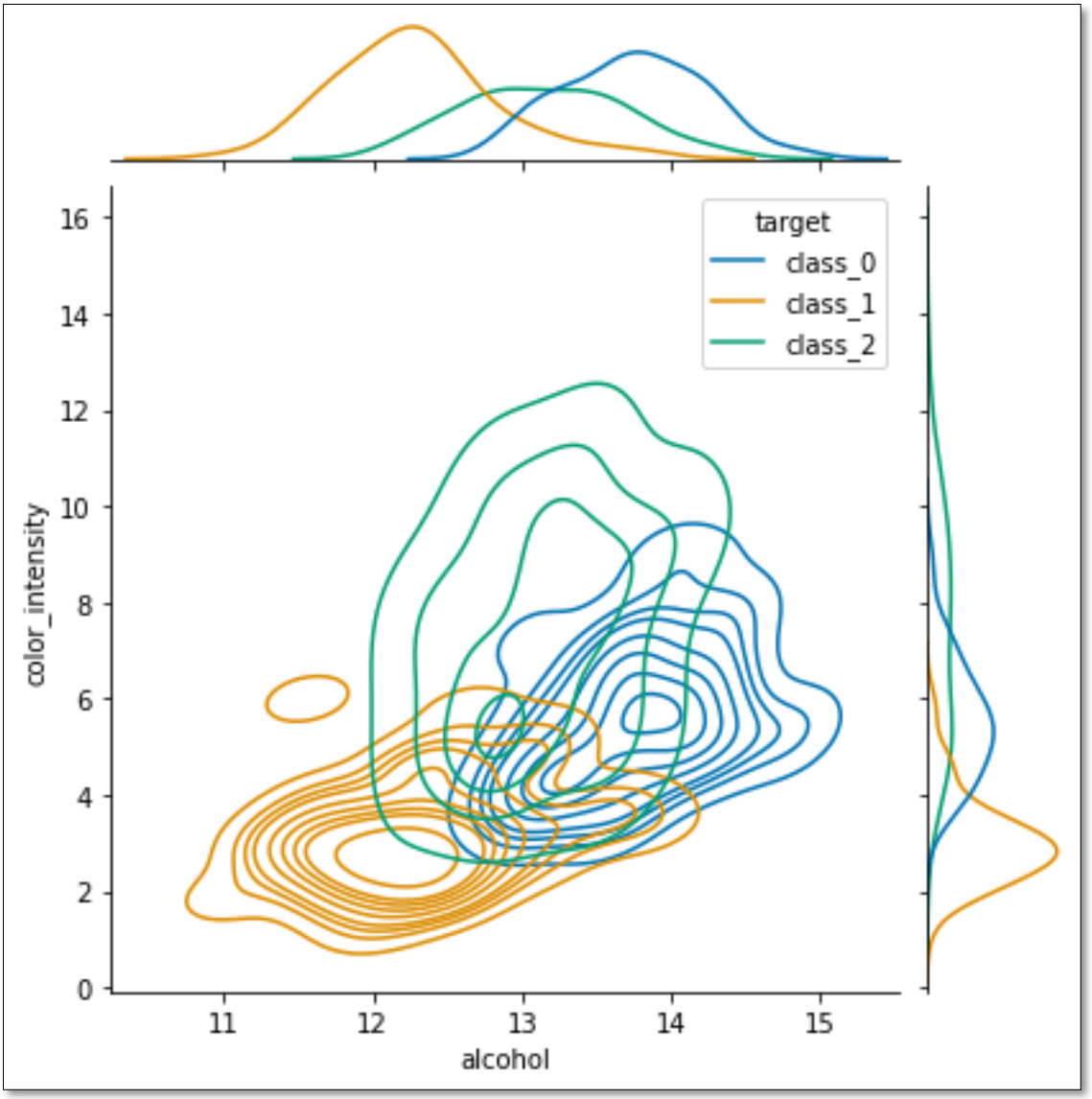


- Adding colours:

```
sns.jointplot(x = 'alcohol', y = 'color_intensity', data = wine,
              hue = "target", palette = my_palette[:3])
```



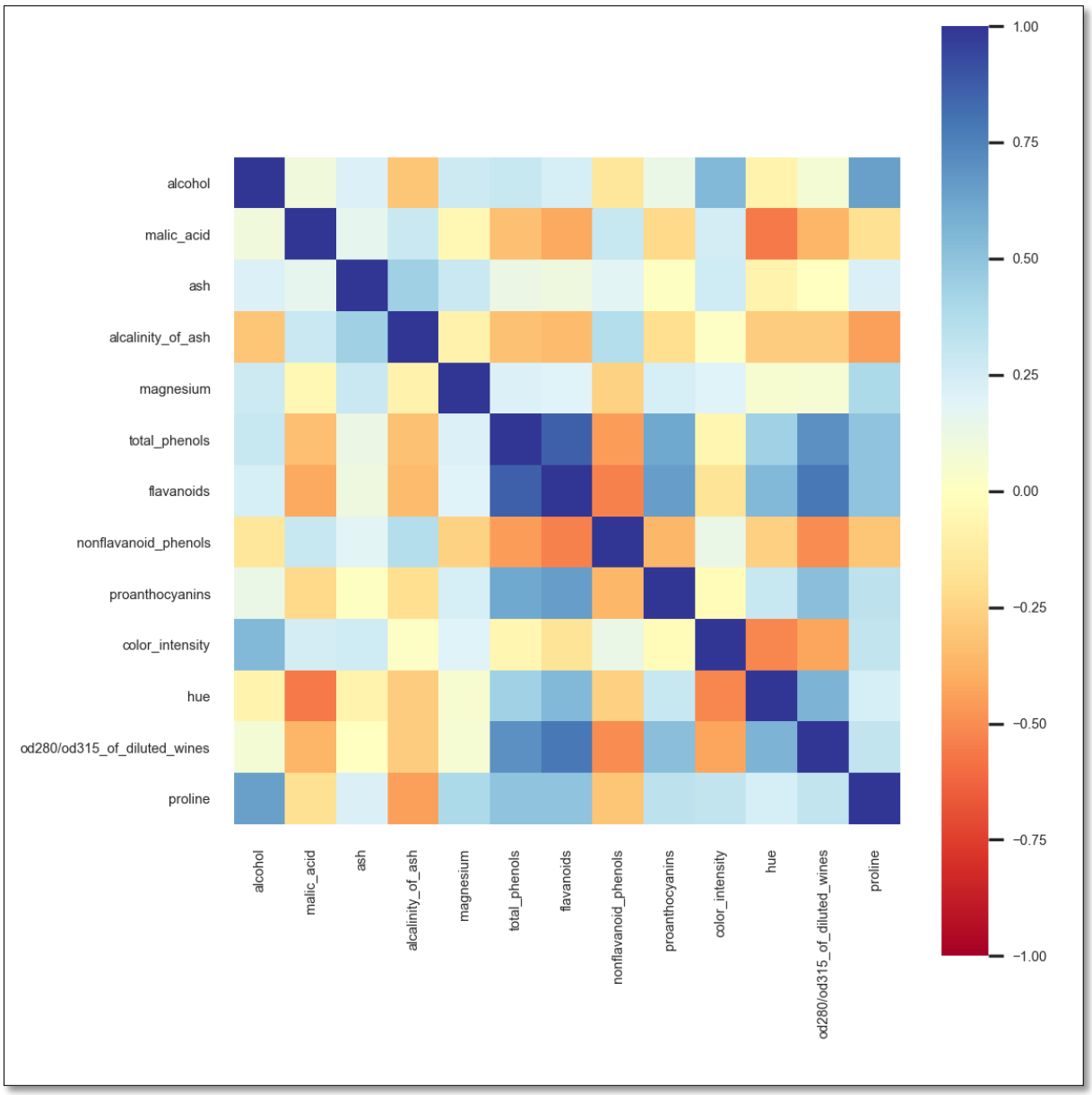

```
sns.jointplot(x = 'alcohol', y = 'color_intensity', data = wine, kind = 'kde',
              hue = "target", palette = my_palette[:3])
```


target
class_0
class_1
class_2
color_intensity
alcohol
0
2
4
6
8
10
12
14
16
11
12
13
14
15

Heat Map

- Heat Map:

```
sns.set(font_scale=0.5)
ax = sns.heatmap(wine.corr(),
                 center= 0, vmin = -1, vmax = 1,
                 cmap= "RdYlBu",     # set the colour of the heatmap
                 square=True)        # set the heatmap to be square shape
ax.figure.tight_layout()          # makes sure all labels are shown in the plot
```



3. Practice Quiz

- Work on *Practice Quiz 09* posted on Canvas.

Useful Resources

-
    - http://