Applied Statistical Methods

Visualization with matplotlib.pyplot and Seaborn

Xuemao Zhang
East Stroudsburg University

February 10, 2023

Outline

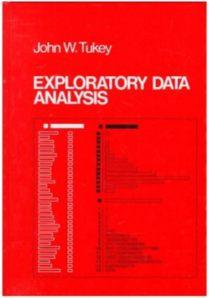
- Introduction to EDA
- Pandas, matplotlib, seaborn and plotly
- Basic plots by Pandas, matplotlib and seaborn
    - Bar charts and Pie charts
    - Histogram
    - ecd plot
    - Box-plot
    - Violin-plot
    - Scatter-plot
    - Scatter-plot matrix
    - Line plot
    - Heat map
- Subplots

Introduction to EDA

- Why do we analyze/summarize data?
  - to understand what has happened or what is happening;
  - to predict what is likely to happen, either in the future or in other circumstances we haven't seen yet;
  - to guide us in making decisions.
- We focus on data visualizations in this section. Predictions will require statistical models.

Introduction to EDA

- John W. Tukey (1977; Exploratory Data Analysis): "The greatest value of a picture is when it forces us to notice what we never expected to see.' '



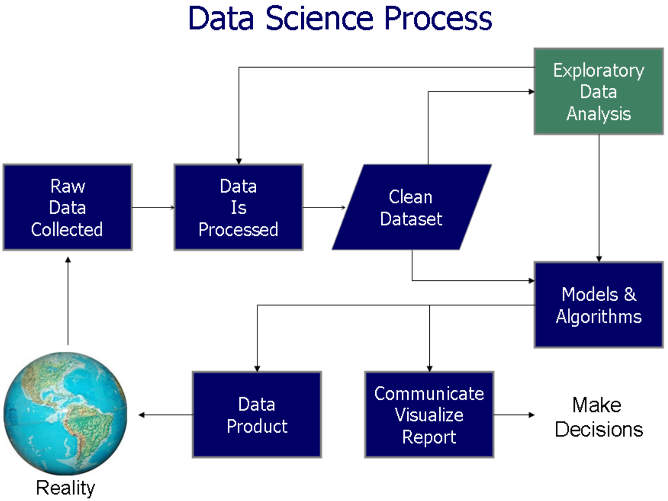
- John W. Tukey coined terms: Boxplot, Stem-and-Leaf plot, ANOVA (Analysis of Variance); "Bit" and "Software".

Introduction to EDA

- The EDA is a statistical approach to make sense of data by using a variety of techniques (mostly graphical). It may help
  - Assess assumption about variables distribution
  - Identify relationship between variables
  - Extract important variables
  - Suggest use of appropriate models
  - Detect problems of collected data (e.g. outliers, missing data, measurement errors)

Introduction to EDA

Data science process flowchart



Data Science Process

Raw Data Collected

Data Is Processed

Clean Dataset

Exploratory Data Analysis

Models & Algorithms

Data Product

Communicate Visualize Report

Make Decisions

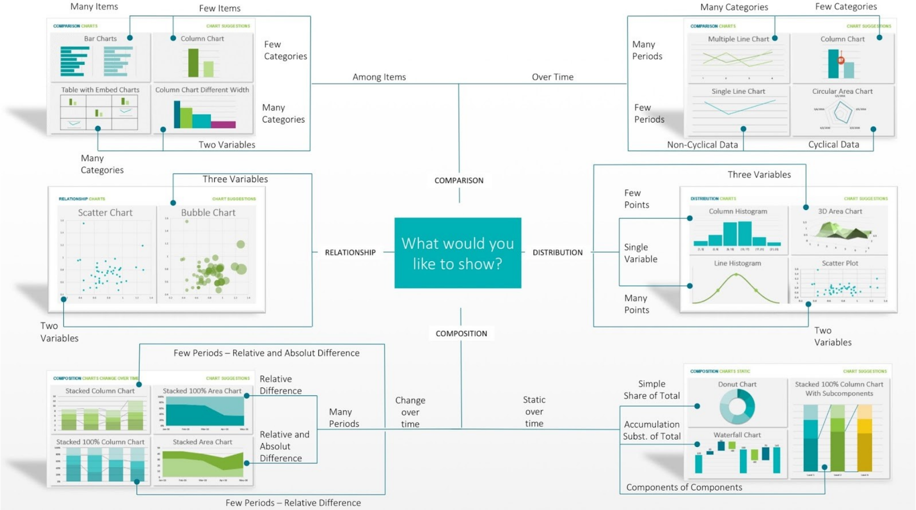
Reality

Introduction to EDA

- **Univarite**
  - Histogram, Stem-and-Leaf, Dot, Q-Q, Density plots
  - Box-and-whisker, Violin
  - Bar, Pie, Polar, Waterfall charts
- **Bivariate**
  - Scatter, Line, Area, Bubble charts
- **Trivariate**
  - 3D Scatter, Contour, Level/Heatmap, Surface plots
- Plots can also be classified as static and interactive visualizations

Introduction to EDA

Which chart to use?



Pandas, matplotlib, seaborn and plotly

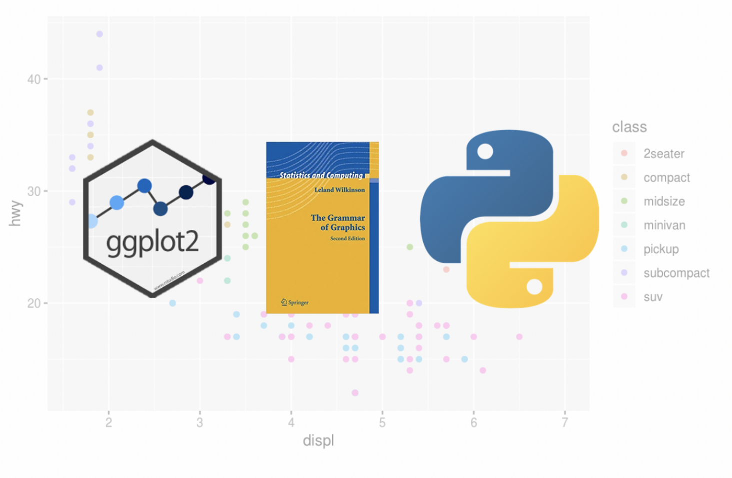
- Pandas comes with built-in plotting functions that rely on Matplotlib.
- Consider Matplotlib on top of pandas if you want to have full control over your visualizations.
- Consider only pandas if you want to organize and rearrange your data to create proof-of-concept visualizations without using other libraries explicitly.
- `pandas.DataFrame.plot`: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html
    - `df.plot.bar()`, `df.plot.pie()`, `df.plot.barh()`, `df.plot.hist()`, `df.plot.line()`, `df.plot.box()`,`df.plot.density()`, `df.plot.area()`,`df.plot.scatter()`, $\cdots$

Pandas, matplotlib, seaborn and plotly

- `matplotlib` was conceived by John Hunter in 2002, originally as a patch to IPython for enabling interactive MATLAB-style plotting via gnuplot from the IPython command line.
  - IPython's creator, Fernando Perez, was at the time scrambling to finish his PhD, and let John know he wouldn't have time to review the patch for several months.
  - John took this as a cue to set out on his own, and the Matplotlib package was born, with version 0.1 released in 2003
- `matplotlib` is a (primarily 2D) desktop plotting package designed for creating publication-quality plots.
- `matplotlib` has a number of add-on toolkits, such as `mplot3d` for 3D plots and basemap for mapping and projections.
- The pyplot module mirrors the MATLAB plotting commands closely. Hence, MATLAB users can easily transit to plotting with Python.
  - It is old-fashioned compared to the package `ggplot2` in R.

Pandas, matplotlib, seaborn and plotly

- `ggplot` now can be used in Python
- A Grammar of Graphics for Python https://plotnine.readthedocs.io/en/stable/



Pandas, matplotlib, seaborn and plotly

- Importing matplotlib
  - The `plt` interface is what we will use most often

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

- Setting Styles: We will use the `plt.style` directive to choose appropriate aesthetic styles for our figures.
  - https://matplotlib.org/stable/api/style_api.html#matplotlib.style.use

```
plt.style.use('classic')
```

```
import matplotlib
matplotlib.style.available
```

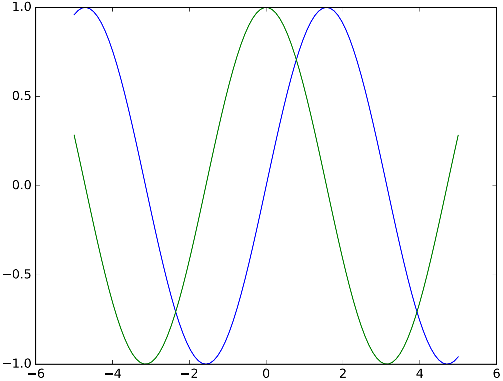
```
## ['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery
```

Pandas, matplotlib, seaborn and plotly

- How to Display Your Plots?
  - If you are using Matplotlib from within a script, the function `plt.show()` is your friend.
  - `plt.show()` starts an event loop, looks for all currently active figure objects, and opens one or more interactive windows that display your figure or figures.
- Matplotlib has multiple figures can be opened, but need to be closed explicitly. `plt.close()` only closes the current figure, `plt.close('all')` would close them all.

Pandas, matplotlib, seaborn and plotly

```
import numpy as np
x = np.linspace(-5, 5, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))  #two plots in a same window
plt.show()
```



Pandas, matplotlib, seaborn and plotly

- Plotly Open Source Graphing https://plotly.com/python/
- `plotly` is a mostly open-source data analytics and visualization tool (with some closed-source products and services).
  - It creates **interactive** charts for web browsers and supports multiple languages, such as Python, Julia, R, and MATLAB.
  - It allows for a high degree of customization.
- Consider `plotly` if you want to display your interactive data visualizations on the web.

Pandas, matplotlib, seaborn and plotly

- seaborn: statistical data visualization https://seaborn.pydata.org/
- `seaborn` is a Python plotting library built on top of `matplotlib`.
  - `seaborn` uses fewer syntax and has easily interesting default themes.
  - Matplotlib works with data frames and arrays.
  - `seaborn` works with the dataset as a whole and is much more intuitive than Matplotlib.
  - `seaborn` is more integrated for working with Pandas data frames. It extends the Matplotlib library for creating beautiful graphics with Python using a more straightforward set of methods.
- Matplotlib is highly customizable and powerful while Seaborn avoids a ton of boilerplate by providing default themes which are commonly used.
- `seaborn` is Matplotlib under the hood. But is specially meant for statistical plotting.

Pandas, matplotlib, seaborn and plotly

- Installation

```
pip install seaborn
```

- Importing seaborn: by convention, Seaborn is imported as sns
- We can set the style by calling Seaborn's `set()` (Alias for `set_theme()`) method.
  - https://seaborn.pydata.org/generated/seaborn.set_theme.html

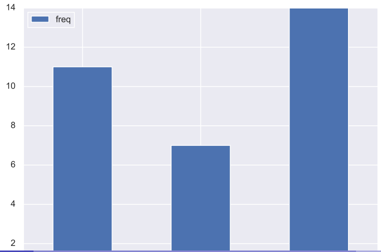
```
import seaborn as sns
sns.set()
```

- it provides high-level commands to create a variety of plot types useful for statistical data exploration

Bar chart with pandas

- Bar chart can be obtained using method `DataFrame.plot.bar()` in **pandas**
  - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html

```
import pandas as pd
mtcars=pd.read_csv("../data/mtcars.csv")
#mtcars['cyl'] = mtcars['cyl'].astype(str)
table=mtcars.groupby(['cyl']).agg(freq=("mpg",'count'))
#table=pd.crosstab(index=mtcars['cyl'],columns='freq')
table.reset_index(inplace=True)
table.plot.bar(x='cyl', y='freq', rot=0)
plt.show()
```



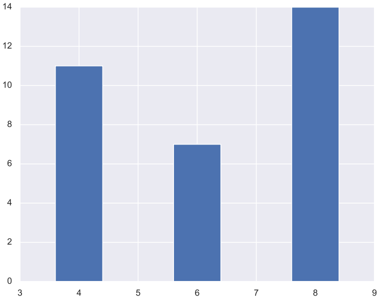
Bar chart with Matplotlib

- Pandas columns may need to be converted to lists using `Series.tolist()`
  - https://pandas.pydata.org/docs/reference/api/pandas.Series.tolist.html

```
#cyl=table.cyl.tolist()
#freq=table.freq.tolist()
plt.bar(table.cyl, table.freq)
```

```
## <BarContainer object of 3 artists>
```

```
plt.show()
```



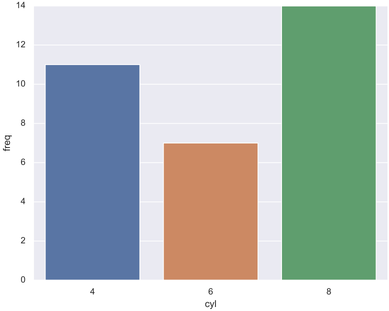
Bar chart with seaborn

- With seaborn, we work with the data frame directly and the syntax is like that of ggplot in R.

```
type(table)
```

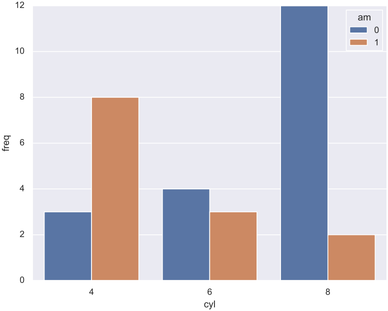
```
## <class 'pandas.core.frame.DataFrame'>
```

```
sns.barplot(data=table, x='cyl',y='freq')
plt.show()
```



Bar chart with seaborn

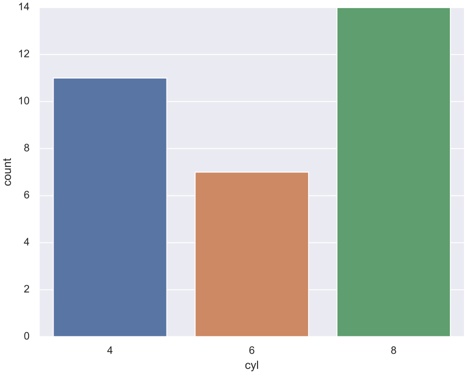
```
table2=mtcars.groupby(['cyl', 'am']).agg(freq=("mpg",'count'))
table2=table2.reset_index()
sns.barplot(x='cyl',y='freq',hue='am',data=table2)
plt.show()
```



Bar chart with seaborn

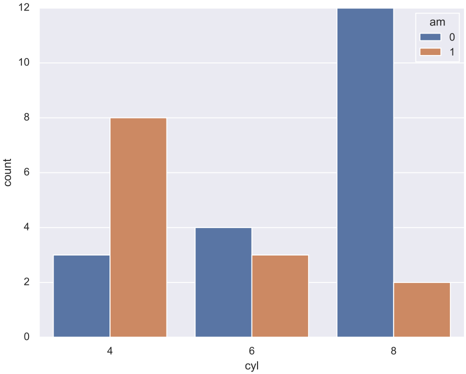
- The easiest way to get a bar plot without counting is to use function `seaborn.countplot()`
  - https://seaborn.pydata.org/generated/seaborn.countplot.html

```
sns.countplot(x=mtcars['cyl'])
plt.show()
```



Bar chart with seaborn

```
sns.countplot(data=mtcars, x='cyl', hue='am')
plt.show()
```



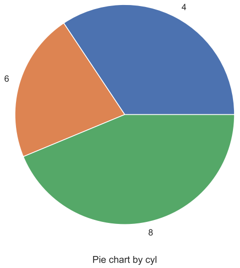
Pie chart with matplotlib

- Pie chart can be produced using function `pyplot.pie()` in `matplotlib`
    - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html

```
plt.pie(table.freq, labels=table.cyl)
```

```
## ([<matplotlib.patches.Wedge object at 0x000002487C9BBF70>, <matplotlib.p
```

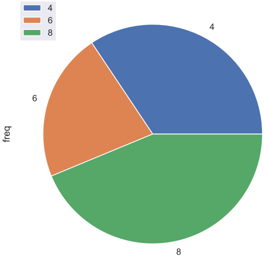
```
plt.xlabel('Pie chart by cyl')
#plt.title("Pie chart by cyl")
plt.show()
```



Pie chart with pandas

- Pie charts are not directly available in Seaborn
- We need to convert the column `cyl` in `table` to index first using method `set_index()`.
  - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.set_index.html

```
table= table.set_index('cyl')
table.plot.pie(y='freq')
plt.show()
```



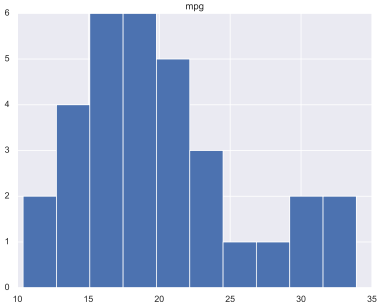
Histogram with pandas

- `pandas.DataFrame.hist`: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.hist.html

```
mtcars.hist(column='mpg')
```

```
## array([[<AxesSubplot: title={'center': 'mpg'}>]], dtype=object)
```

```
plt.show()
```



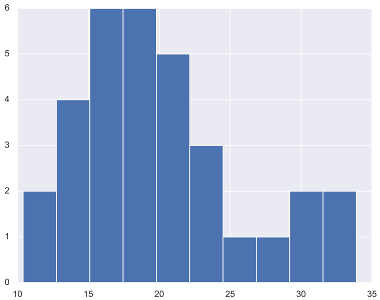
Histogram with matplotlib

- https://matplotlib.org/stable/plot_types/stats/hist_plot.html#

```
plt.hist(mtcars.mpg)
```

```
## (array([2., 4., 6., 6., 5., 3., 1., 1., 2., 2.]), array([10.4 , 12.75, 1
##         31.55, 33.9 ]), <BarContainer object of 10 artists>)
```

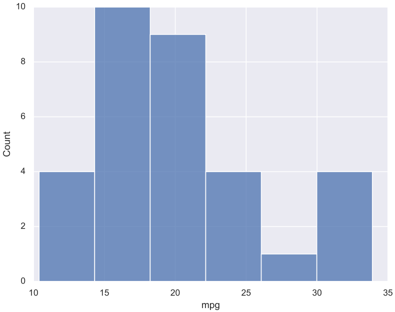
```
plt.show()
```



Histogram with seaborn

- Histogram by function `seaborn.histplot()`: https://seaborn.pydata.org/generated/seaborn.histplot.html

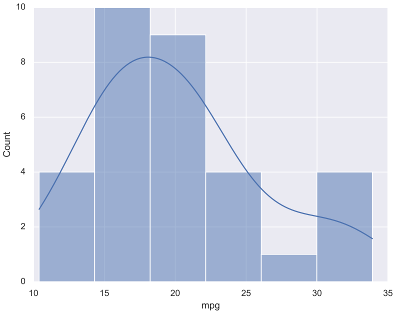
```
sns.histplot(data=mtcars,x='mpg')
plt.show()
```



Histogram

- The technique of Kernel-Density-Estimation (KDE) can be used to estimate the smooth **probability density curve** of a data set.
- Add a kernel density estimate to smooth the histogram, providing complementary information about the shape of the distribution.
- Hist and kde with seaborn:

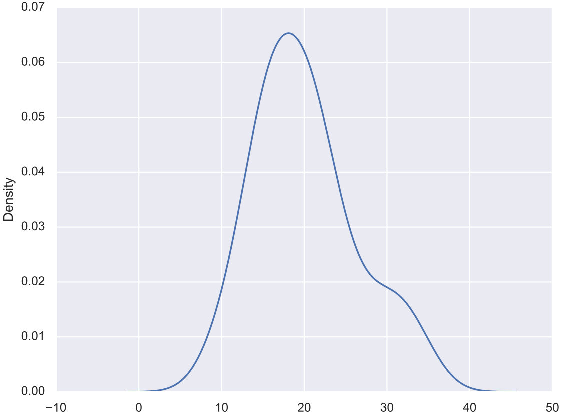
```
sns.histplot(data=mtcars,x='mpg',kde=True)
plt.show()
```



Histogram

- KDE plot only by pandas

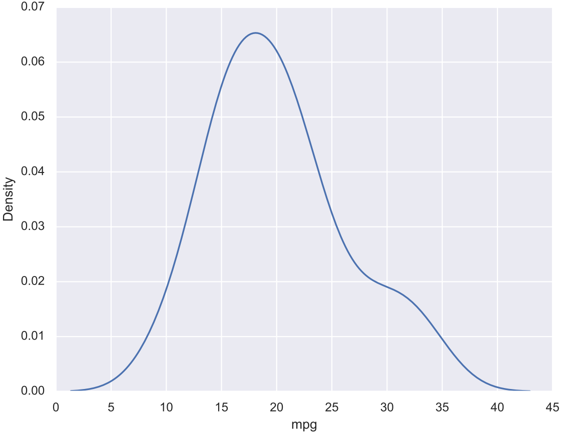
```
mtcars['mpg'].plot.kde()
```



Histogram

- KDE plot only with function `seaborn.kdeplot()`: https://seaborn.pydata.org/generated/seaborn.kdeplot.html

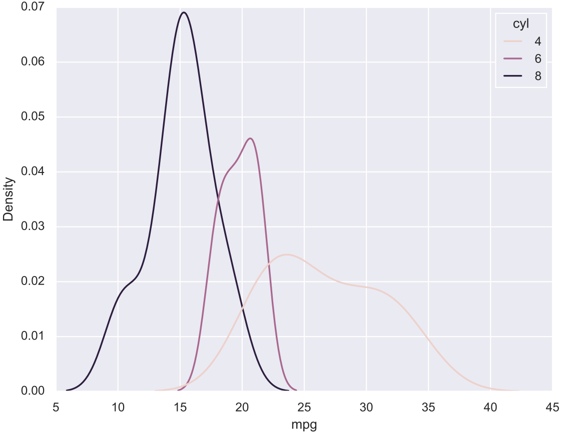
```
sns.kdeplot(data=mtcars,x='mpg')
plt.show()
```



Histogram

- KDE plot only with function `seaborn.kdeplot()`: https://seaborn.pydata.org/generated/seaborn.kdeplot.html

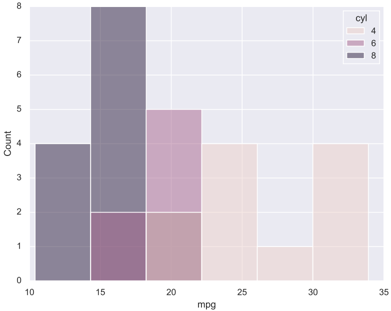
```
sns.kdeplot(data=mtcars,x='mpg', hue='cyl')
plt.show()
```



Histogram

- Layers: we can draw multiple histograms wrt a categorical variable with hue mapping

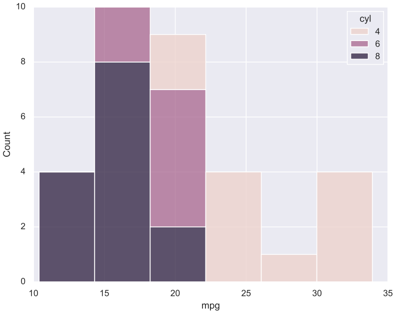
```
sns.histplot(data=mtcars,x='mpg',hue="cyl")
plt.show()
```



Histogram

- Stacks: The default approach to plotting multiple distributions is to “layer” them, but you can also `stack` them

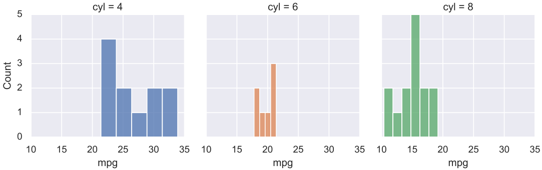
```
sns.histplot(data=mtcars,x='mpg',hue="cyl",multiple="stack")
plt.show()
```



Histogram

- **Faceting**: Facets divide a plot into subplots based on the values of one or more categorical variables.
- `seaborn.FacetGrid()`: https://seaborn.pydata.org/generated/seaborn.FacetGrid.html
  - `sns.FacetGrid()` constructs a grid
  - To draw a plot on every facet, pass a function and the name of one or more columns in the dataframe to `FacetGrid.map_dataframe()`

```
g=sns.FacetGrid(data=mtcars, col='cyl', hue='cyl') #1 by 3 grid
g.map_dataframe(sns.histplot, x="mpg")
```

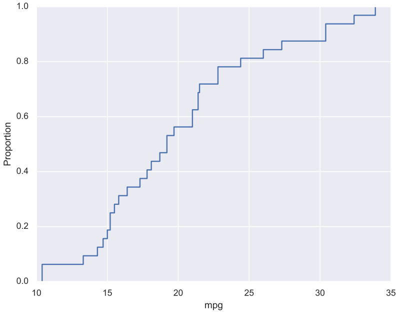


```
plt.close()
```

ecd plot with seaborn

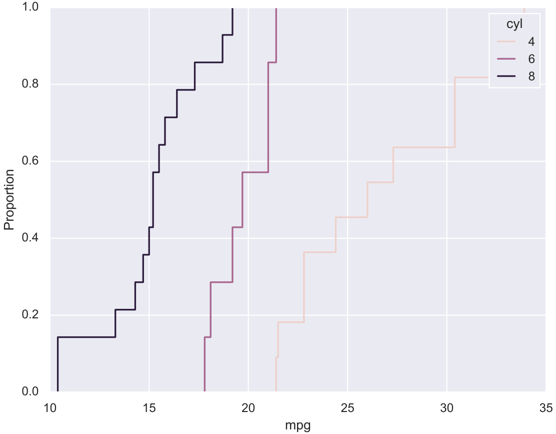
- A cumulative frequency curve indicates the number (or percent) of data with less than a given value.
- Function `seaborn.ecdfplot()` plots empirical cumulative distribution functions.
  - https://seaborn.pydata.org/generated/seaborn.ecdfplot.html

```
sns.ecdfplot(data=mtcars, x="mpg")
plt.show()
```



ecd plot with seaborn

```
sns.ecdfplot(data=mtcars, x="mpg", hue='cyl')
plt.show()
```

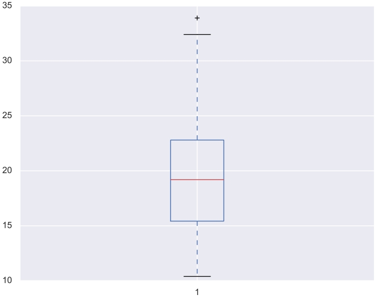


Box plot with matplotlib

```
plt.boxplot(mtcars['mpg'])
```

```
## {'whiskers': [<matplotlib.lines.Line2D object at 0x000002487C913D90>, <m
```

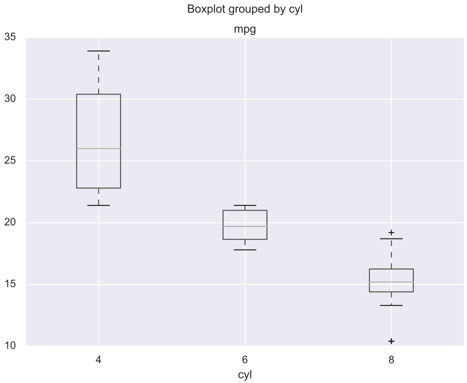
```
plt.show()
```



Box plot with pandas

- Box plots are frequently used in scientific publications to indicate data values in two or more groups.

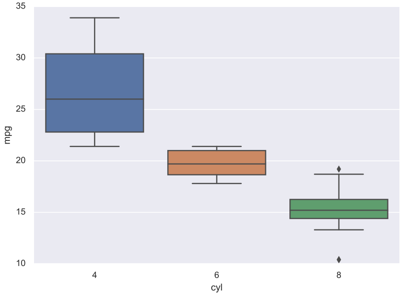
```
mtcars.boxplot(column='mpg', by="cyl")
plt.show()
```



Box plot with seaborn

- https://seaborn.pydata.org/generated/seaborn.boxplot.html

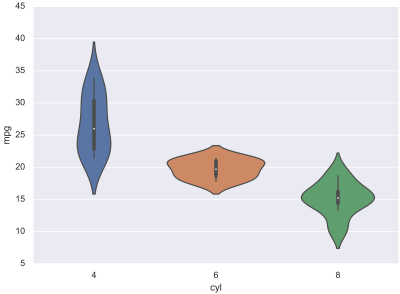
```
sns.boxplot(data=mtcars, x="cyl", y='mpg')
plt.show()
```



Violin Plot with seaborn

- Boxplot can be combined with KDE-plots to produce the so-called violin plots, where the vertical axis is the same as for the box-plot, but in addition a KDE-plot is shown symmetically along the horizontal direction.
- https://seaborn.pydata.org/generated/seaborn.violinplot.html

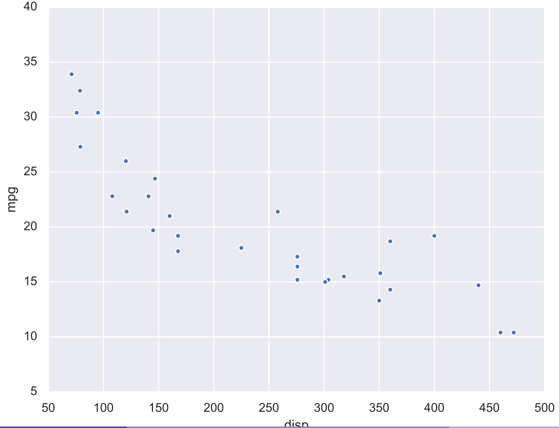
```
sns.violinplot(data=mtcars, x="cyl", y='mpg')
plt.show()
```



Scatter-plot with pandas

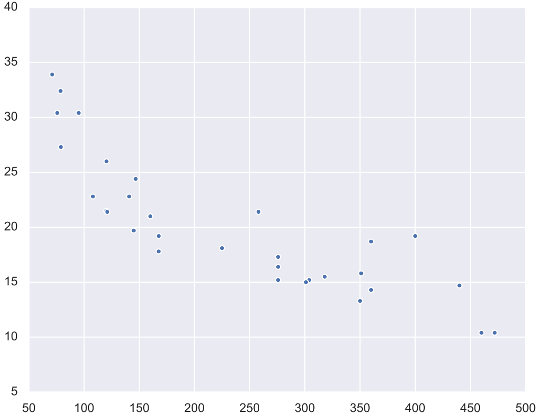
- Bivariate scatter plots tell us the relationship between two numerical variables.
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.scatter.html

```
mtcars.plot.scatter(x="disp", y='mpg')
plt.show()
```



Scatter-plot with matplotlib

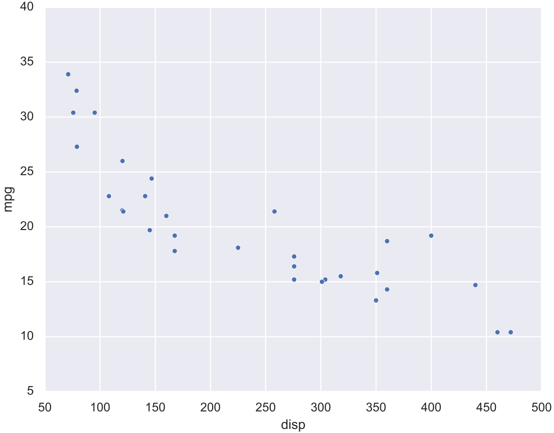
```
plt.scatter(x=mtcars.disp, y=mtcars.mpg)
plt.show()
```



Scatter-plot with seaborn

- https://seaborn.pydata.org/generated/seaborn.scatterplot.html

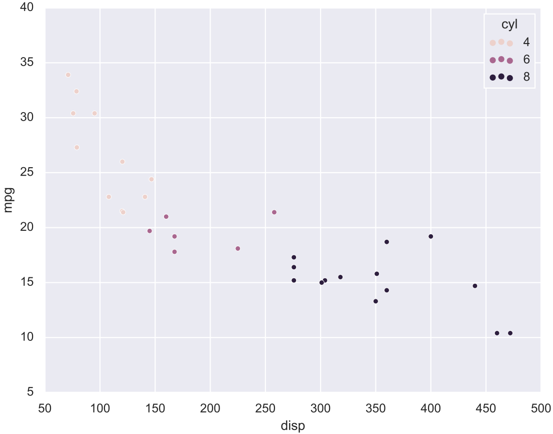
```
sns.scatterplot(data=mtcars, x="disp", y='mpg')
plt.show()
```



Scatter-plot with seaborn

- https://seaborn.pydata.org/generated/seaborn.scatterplot.html

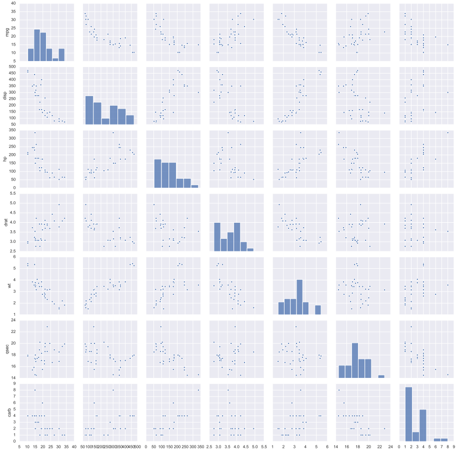
```
sns.scatterplot(data=mtcars, x="disp", y='mpg', hue='cyl')
plt.show()
```



Scatter-plot matrix with seaborn

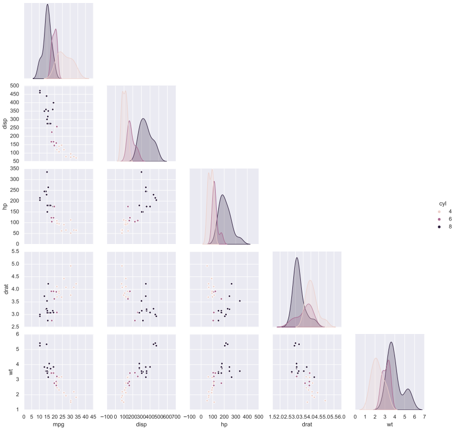
- We use scatter-plot matrix to check the relationships among several numerical variables.
  - https://seaborn.pydata.org/generated/seaborn.pairplot.html

```
mtcars1=mtcars.drop(['Unnamed: 0', 'cyl', 'vs','am','gear'],axis=1)
sns.pairplot(mtcars1)
```

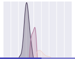


Scatter-plot matrix with seaborn

```
mtcars2=mtcars[['mpg', 'cyl', 'disp', 'hp', 'drat', 'wt']]
sns.pairplot(mtcars2, corner=True, hue='cyl')
```



```
plt.show()
```



Scatter-plot matrix with pandas

```
pd.plotting.scatter_matrix(mtcars2, alpha=0.2)
```

```
## array([[<AxesSubplot: xlabel='mpg', ylabel='mpg'>,
##          <AxesSubplot: xlabel='cyl', ylabel='mpg'>,
##          <AxesSubplot: xlabel='disp', ylabel='mpg'>,
##          <AxesSubplot: xlabel='hp', ylabel='mpg'>,
##          <AxesSubplot: xlabel='drat', ylabel='mpg'>,
##          <AxesSubplot: xlabel='wt', ylabel='mpg'>],
##         [<AxesSubplot: xlabel='mpg', ylabel='cyl'>,
##          <AxesSubplot: xlabel='cyl', ylabel='cyl'>,
##          <AxesSubplot: xlabel='disp', ylabel='cyl'>,
##          <AxesSubplot: xlabel='hp', ylabel='cyl'>,
##          <AxesSubplot: xlabel='drat', ylabel='cyl'>,
##          <AxesSubplot: xlabel='wt', ylabel='cyl'>],
##         [<AxesSubplot: xlabel='mpg', ylabel='disp'>,
##          <AxesSubplot: xlabel='cyl', ylabel='disp'>,
##          <AxesSubplot: xlabel='disp', ylabel='disp'>,
##          <AxesSubplot: xlabel='hp', ylabel='disp'>,
##          <AxesSubplot: xlabel='drat', ylabel='disp'>,
##          <AxesSubplot: xlabel='wt', ylabel='disp'>],
##         [<AxesSubplot: xlabel='mpg', ylabel='hp'>,
##          <AxesSubplot: xlabel='cyl', ylabel='hp'>,
```

Line plot with seaborn

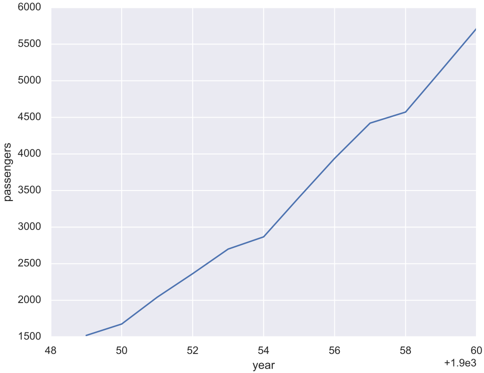
- A graph can be a powerful vehicle for displaying change over time. The most common time-dependent graph is the time series line graph.
  - A **time series** is a set of quantitative values obtained at successive time points. The intervals between time points (e.g., hours, days, weeks, months, or years) are usually equal.
- https://seaborn.pydata.org/generated/seaborn.lineplot.html

```
flights = sns.load_dataset("flights")
flights.head()
```

```
##     year month  passengers
## 0   1949   Jan         112
## 1   1949   Feb         118
## 2   1949   Mar         132
## 3   1949   Apr         129
## 4   1949   May         121
```

Line plot with seaborn

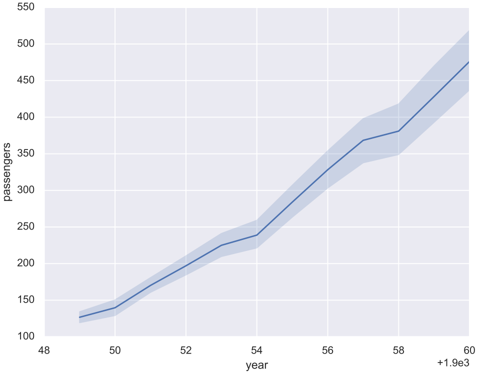
```
flights_year=flights.groupby(['year']).agg(passengers=("passengers",'sum'))
sns.lineplot(data=flights_year, x="year", y="passengers")
plt.show()
```



Line plot with seaborn

- By default, `sns.lineplot` will aggregate over repeated values (each year) to show the **mean** and 95% confidence interval.

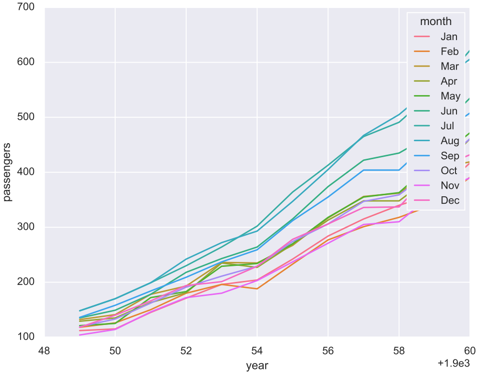
```
sns.lineplot(data=flights, x="year", y="passengers")
plt.show()
```



Line plot with seaborn

- Line plot by a categorical variable

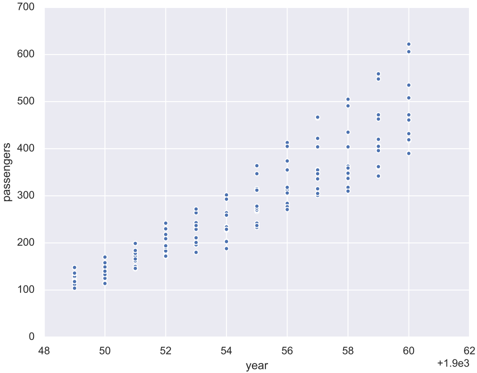
```
sns.lineplot(data=flights, x="year", y="passengers", hue="month")
plt.show()
```



Line plot with pandas

- The data is not time-series

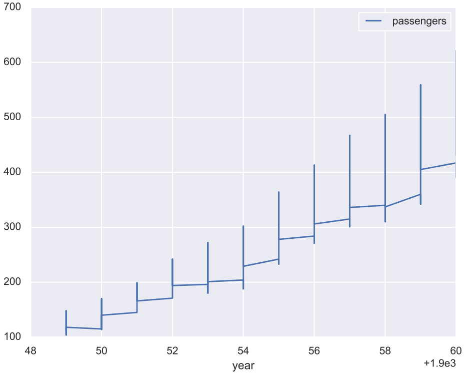
```
flights.plot.scatter(x="year", y="passengers")
plt.show()
```



Line plot with pandas

- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.line.html

```
flights.plot.line(x="year", y="passengers")
plt.show()
```



Heat map

- Heat maps is a very useful graphical tool to better understand or present data stored in **matrix** forms.
- A **heatmap** is basically a table that has colors in place of numbers.
- Colors correspond to the level of the measurement.
- It is quite straight forward to make a heat map, as shown on the examples in this lecture. However be careful to understand the underlying mechanisms.
- You might prefer to conduct **cluster analysis** and then permute the rows and the columns of the matrix to place similar values near each other according to the clustering. Cluster analysis will be discussed later in this course.

Heat map

- Let's start with a very simple matrix

```
data=pd.DataFrame({"x1":[1,3,2],"x2":[2,15,8]},
index=[1,2,3])
print(data)
```

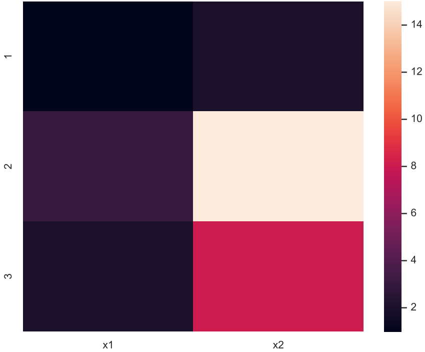
```
##     x1  x2
## 1    1   2
## 2    3  15
## 3    2   8
```

Heat map

- Next, we can prepare a basic heat map. In the heat map in the next slide,
  - The x axis represents columns in matrix. The first column is on the left (the lowest value on the axis), the second column is on the right (analogously - the highest value).
  - The y axis represents rows and the first row is on the bottom.
  - By default, red colour represents the highest values in our matrix, while the lowest are darker.

Heat map

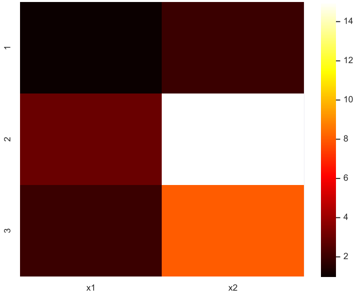
```
sns.heatmap(data)
plt.show()
```



Heat map

- Select a different colormap by name https://matplotlib.org/stable/tutorials/colors/colormaps.html

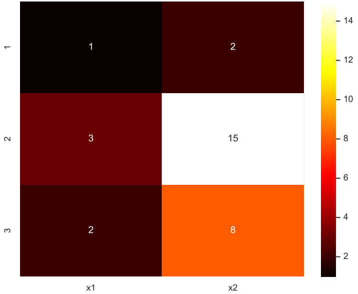
```
sns.heatmap(data, cmap="hot")
plt.show()
```



Heat map

- annot=True write the data value in each cell.

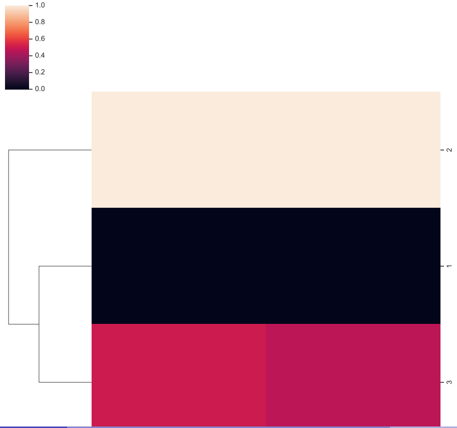
```
sns.heatmap(data, cmap="hot", annot=True)
plt.show()
```



Heat map

- seaborn.clustermap(): Plot a matrix using hierarchical clustering to arrange the rows and columns.
  - https://seaborn.pydata.org/generated/seaborn.clustermap.html
    - standard_scale=1 standardize the columns
    - row_cluster=True group the rows

```
sns.clustermap(data, standard_scale=1,row_cluster=True, col_cluster=False)
```



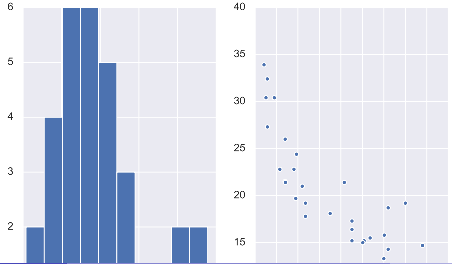
Subplots with matplotlib

- With the `subplot()` function you can draw multiple plots in one figure.
  - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

```
import matplotlib.pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.hist(mtcars.mpg)
```

```
## (array([2., 4., 6., 6., 5., 3., 1., 1., 2., 2.]), array([10.4 , 12.75, 1
##         31.55, 33.9 ]), <BarContainer object of 10 artists>)
```

```
ax2.scatter(x=mtcars.disp, y=mtcars.mpg)
plt.show()
```



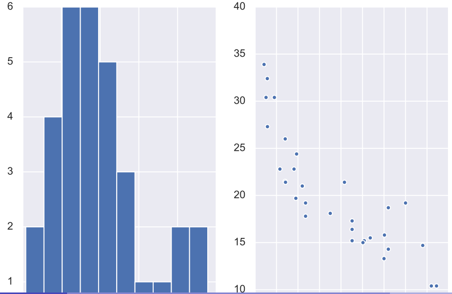
Subplots with matplotlib

- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1, 2)
axs[0].hist(mtcars.mpg)
```

```
## (array([2., 4., 6., 6., 5., 3., 1., 1., 2., 2.]), array([10.4 , 12.75, 1
##          31.55, 33.9 ]), <BarContainer object of 10 artists>)
```

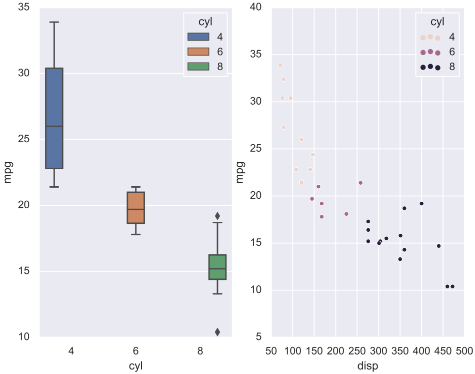
```
axs[1].scatter(x=mtcars.disp, y=mtcars.mpg)
plt.show()
```



Subplots with seaborn

- To get subplots with seaborn, we still use an array of Axes set by matplotlib

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1, 2)
sns.boxplot(data=mtcars, x="cyl", y='mpg', hue='cyl', ax=axs[0])
sns.scatterplot(data=mtcars, x="disp", y='mpg', hue='cyl',ax=axs[1])
plt.show()
```



License