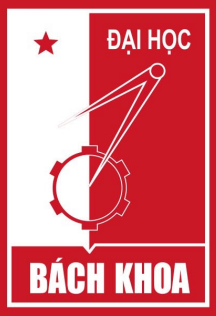


HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Introduction to
Data Science
(IT4142E)

Contents

- Lecture 1: Overview of Data Science
- Lecture 2: Data crawling and preprocessing
- Lecture 3: Data cleaning and integration
- Lecture 4: Exploratory data analysis
- **Lecture 5: Data visualization**
- Lecture 6: Multivariate data visualization
- Lecture 7: Machine learning
- Lecture 8: Big data analysis
- Lecture 9: Capstone Project guidance
- Lecture 10+11: Text, image, graph analysis
- Lecture 12: Evaluation of analysis results

This lecture

1. How to choose the right chart?
2. Bar Chart – Column Chart
3. Line Chart
4. Histogram
5. Scatter Plot
6. Violin
7. Other charts
8. Multivariable Visualization

1. How to choose the right chart?

- Data visualization is a technique to communicate insights from data through visual representation
- Main goal: is to distill large datasets into visual graphics to allow for a straighforward understanding of complex relationship within the data
- It is important to choose the right chart for visualizing your data

What story do you want to tell?

- It is important to understand why we need a kind of chart
  - Graphs
  - Plots
  - Maps
  - Diagrams
  - ...
- Relationship
- Data over time
- Ranking
- Distribution
- Comparison

Relationship

- To display a connection or correlation between two or more variables
- When assessing a relationship between data sets, we are trying to understand how these data sets combine and interact with each other
- The relationship or correlation can be positive or negative
  - Whether or not the variables might be supportive or working against each other

Relationship

- Scatter plot
- Histogram
- Pair Plot
- Heat map

Data over time

- Goal: to explore the relationship between variables to find trends or changes over time
- The date/time appears as a link property between variables, so a kind of relationship
- Line chart
- Area chart
- Stack Area Chart
- Area Chart Unstacked

Ranking

- Goal: to display the relative order of data values
- Vertical bar chart
- Horizontal bar chart or Column Chart
- Multi-set bar chart
- Stack bar chart
- Lollipop Chart

Distribution

- Goal: to see how a variable is distributed
- Histogram
- Density Curve with Histogram
- Density plot
- Box plot
- Strip plot
- Violin Plot
- Population Pyramid

Comparison

- Goal: to display the trends between multiple variable in datasets or multiple categories within a single variable
- Bubble chart
- Bullet chart
- Pie chart
- Net pie chart
- Donut chart
- TreeMap
- Diverging bar
- Choropleth map
- Bubble map

2. Bar/Column Chart

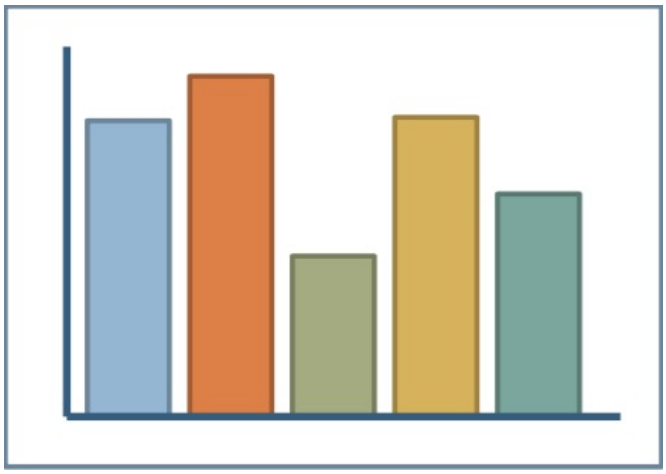
- A series of bars illustrating a variable's development
- 4 types of bar charts:
  - Horizontal bar chart
  - Vertical bar chart
  - Group bar chart
  - Stacked bar chart
- This kind of chart is appropriated when we want to track the development of one or two variables over time
- One axis shows the specific categories being compared (independent variable)
- The other axis represents a measured value (dependent variable)

Vertical Bar Chart (Column Chart)

- Distinguish it from histograms
  - not to display a continuous developments over an interval
  - discrete data
  - data is categorical and used to answer the question of **how many** in each category
- Used to compare several items in a specific range of values
- Ideal for comparing a single category of data between individual sub-items

Vertical Bar Chart (Column Chart)

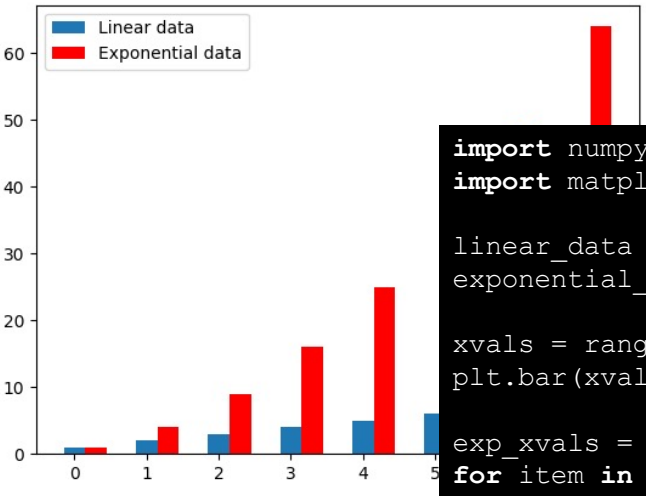
Quantitative
Dependent
variable



Benefits from both position (top of bar) and length (size of bar)

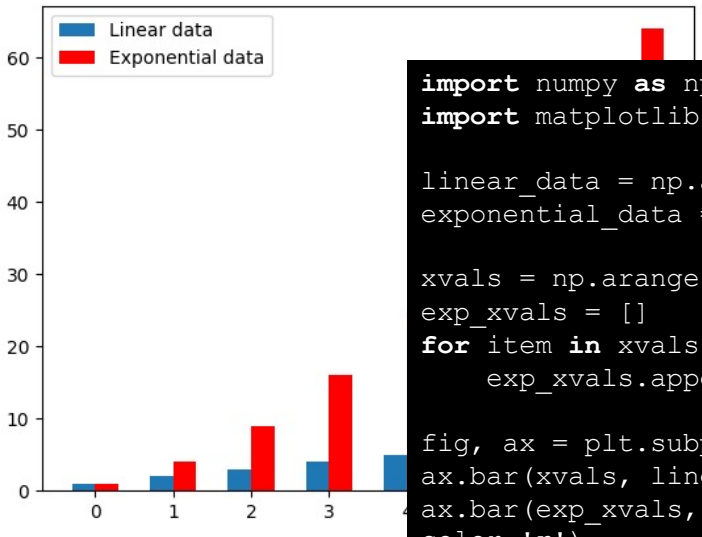
Discrete/Nominal
Independent variable

Vertical Bar Chart (Column Chart)




```
import numpy as np
import matplotlib.pyplot as plt

linear_data = np.array([1, 2, 3, 4, 5, 6, 7, 8])
exponential_data = linear_data ** 2

xvals = range(len(linear_data))
plt.bar(xvals, linear_data, width=0.3)

exp_xvals = []
for item in xvals:
    exp_xvals.append(item+0.3)
plt.bar(exp_xvals, exponential_data, width=0.3,
color='r')

plt.legend(['Linear data', 'Exponential data'])
plt.show()
```

Vertical Bar Chart (Column Chart)




```
import numpy as np
import matplotlib.pyplot as plt

linear_data = np.array([1, 2, 3, 4, 5, 6, 7, 8])
exponential_data = linear_data ** 2

xvals = np.arange(len(linear_data))
exp_xvals = []
for item in xvals:
    exp_xvals.append(item+0.3)

fig, ax = plt.subplots()
ax.bar(xvals, linear_data, width=0.3)
ax.bar(exp_xvals, exponential_data, width=0.3,
color='r')
ax.legend(['Linear data', 'Exponential data'])
ax.set_xticks(xvals + 0.3 / 2)
ax.set_xticklabels(xvals)
plt.show()
```

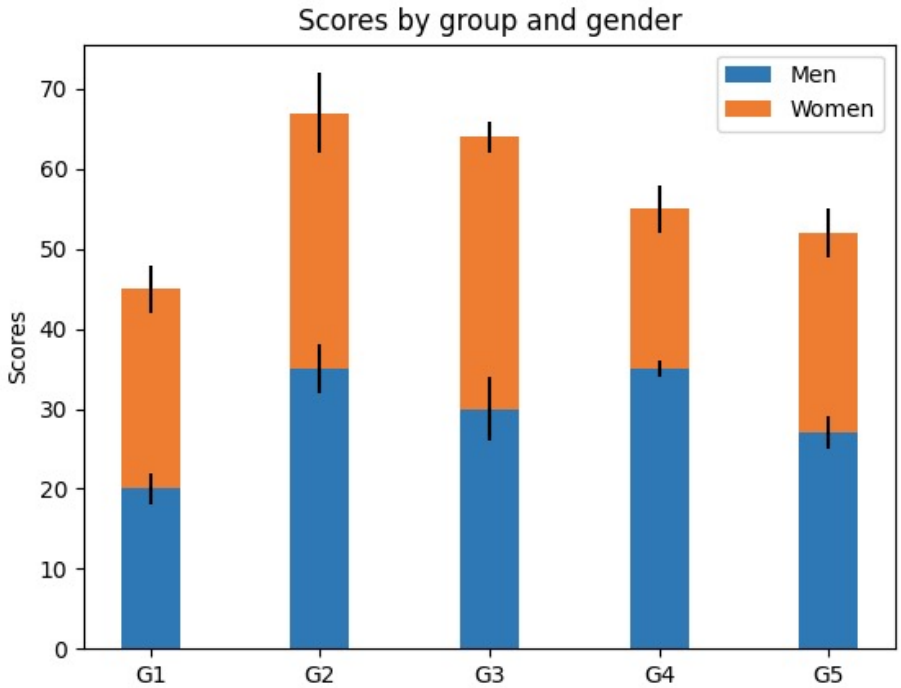

Horizontal Bar Chart

- Represent the data horizontally
- The data categories are shown on the y-axis
- The data values are shown on the x-axis
- The length of each bar is equal to the value corresponding to the data category
- All bars go across from left to right
- Use `barh()` function

Stacked Bar Chart

- Stacked bar charts segment their bars
- Used to show how a broader category is divided into smaller categories
- The relationship of each part on the total amount is also showed
- Place each value for the segment after the previous one
- The total value of the bar chart is all the segment values added together
- Ideal for comparing the total amount across each group/segmented bar

Stacked Bar Chart



Stacked Bar Chart

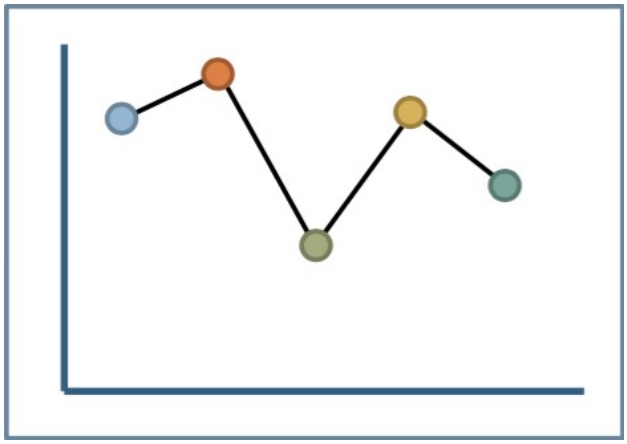
```
import matplotlib.pyplot as plt


labels = ['G1', 'G2', 'G3', 'G4', 'G5']
men_means = [20, 35, 30, 35, 27]
women_means = [25, 32, 34, 20, 25]
men_std = [2, 3, 4, 1, 2]
women_std = [3, 5, 2, 3, 3]
width = 0.35       # the width of the bars: can also be len(x) sequence

fig, ax = plt.subplots()

ax.bar(labels, men_means, width, yerr=men_std, label='Men')
ax.bar(labels, women_means, width, yerr=women_std, bottom=men_means,
       label='Women')

ax.set_ylabel('Scores')
ax.set_title('Scores by group and gender')
ax.legend()

plt.show()
```


3. Line Chart

- Line charts are used to display quantitative values over a continuous interval or period
- Drawn by first plotting data points on a cartesian coordinate grid and then connecting them
- Y-axis has a quantitative value
- X-axis is a timescale or a sequence of intervals
- Best for continuous data
- Most frequently used to show trends and analyze how the data has changed over time

Line charts

Quantitative continuous dependent variable



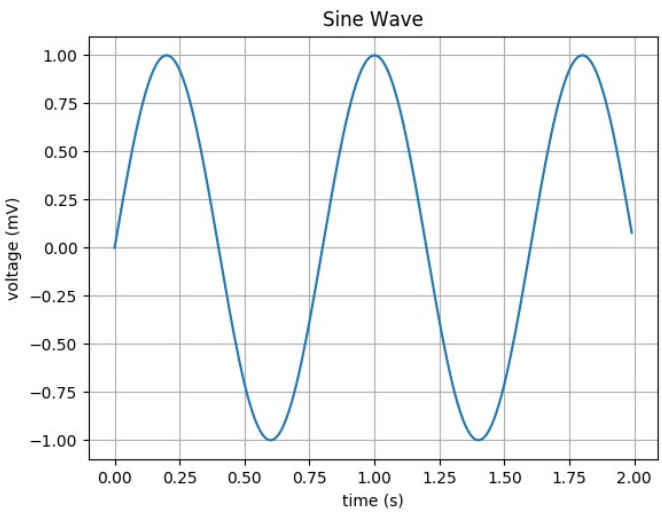
Benefits from position but not length

Quantitative continuous independent variable

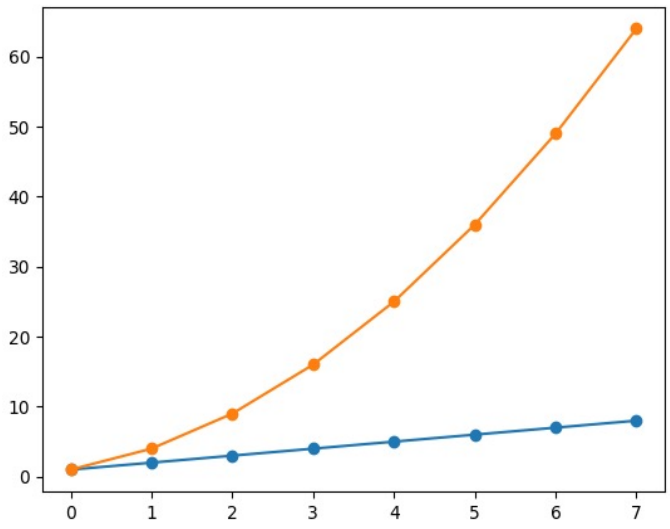
Line chart (pylab vs pyplot

```
from pylab import *
t = arange(0.0, 2.0, 0.01)
s = sin(2.5*pi*t)
plot(t,s)

xlabel('time (s)')
ylabel('voltage (mV)')
title('Sine Wave')
grid(True)
show()
```

```
import numpy as np
import matplotlib.pyplot as plt
t = np.arange(0.0, 2.0, 0.01)
s = np.sin(2.5*np.pi*t)
plt.plot(t,s)

plt.xlabel('time (s)')
plt.ylabel('voltage (mV)')
plt.title('Sine Wave')
plt.grid(True)
plt.show()
```

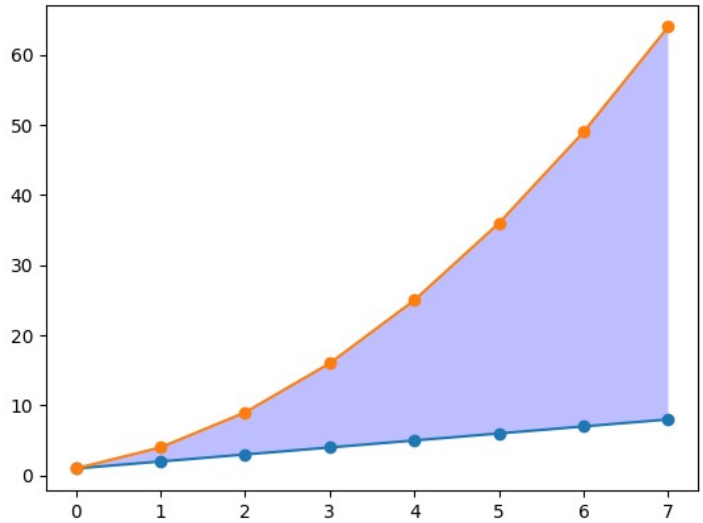


Line chart (cont.)




```
import numpy as np
import matplotlib.pyplot as plt
linear_data =
np.array([1,2,3,4,5,6,7,8])
exponential_data =
linear_data**2
plt.plot(linear_data, '-o',
exponential_data, '-o')
plt.show()
```


Line chart (cont.)

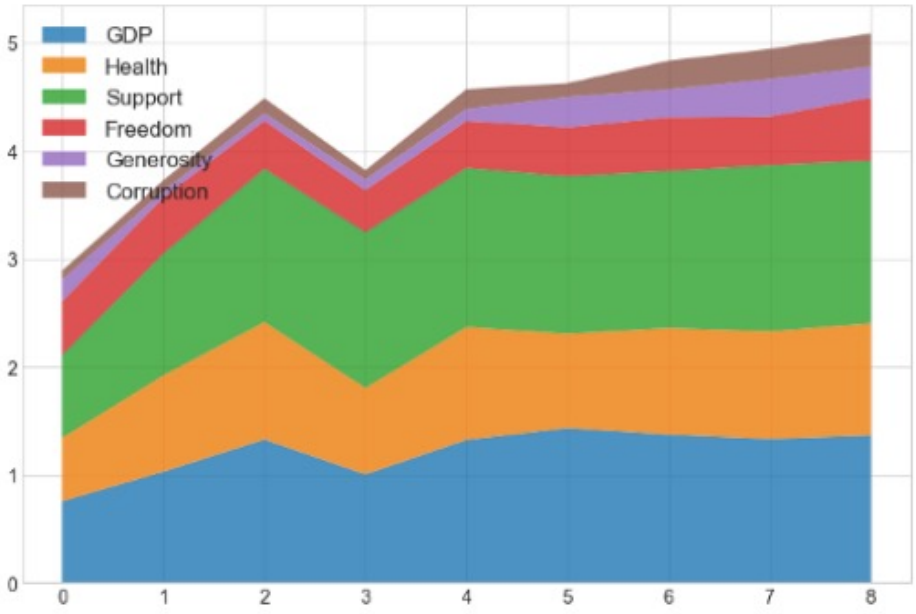



```
import numpy as np
import matplotlib.pyplot as
plt
linear_data =
np.array([1,2,3,4,5,6,7,8])
exponential_data =
linear_data**2
plt.plot(linear_data, '-o',
exponential_data, '-o')
plt.gca().fill_between(range(l
en(linear_data)),

linear_data, exponential_data,

facecolor='blue',

alpha=0.25)
plt.show()
```

Area Chart

- Built based on line chart
- The area between the x-axis and the line is filled in with color or shading
- Ideal for clearly illustrating the magnitude of change between two or more data points
- Use stackplot() function
- Or just fill in color the area between two lines

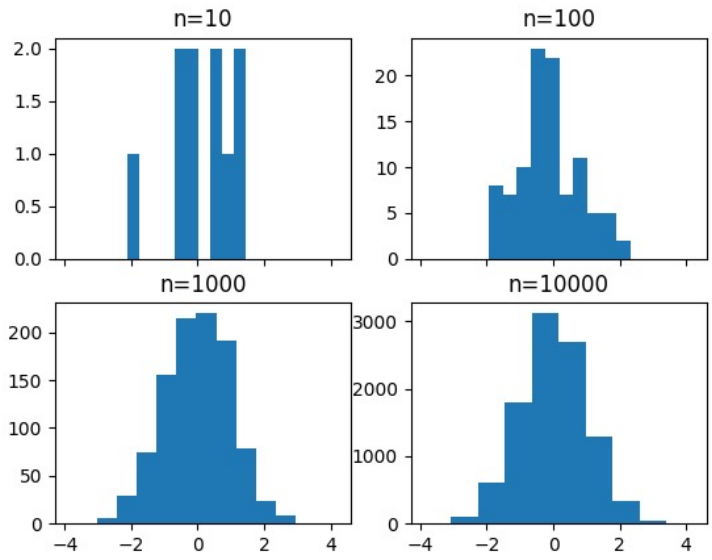
Area Chart



4. Histogram

- Histogram is an accurate representation of the distribution of numerical data
- An estimation of the probability distribution of a continuos variable
- To construct a histogram, follow these steps
  - Bin the range of values
  - Divide the entire range of values into a series of intervals
  - Count how many values fall into each interval
- Bins are usually specified as consecutive, non-overlapping intervals of variable

Histogram example



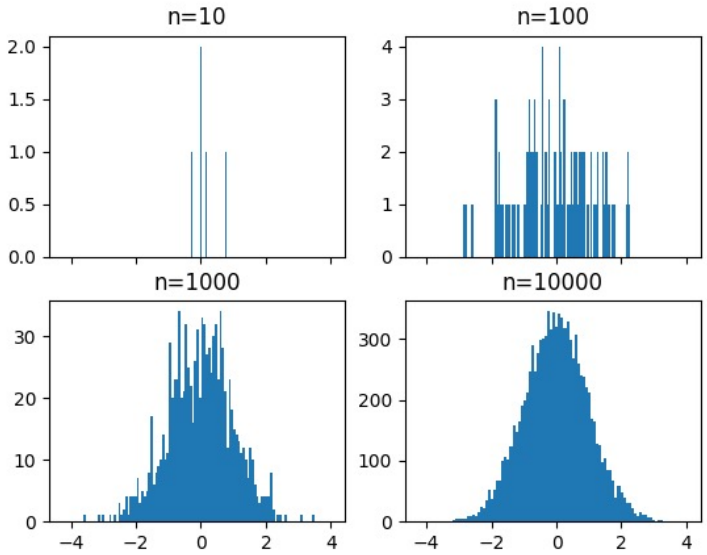
Histogram example

```
import numpy as np
import matplotlib.pyplot as plt

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,
    sharex=True)
axs = [ax1, ax2, ax3, ax4]

for n in range(0, len(axs)):
    sample_size = 10**(n+1)
    sample = np.random.normal(loc=0.0, scale=1.0,
        size=sample_size)
    axs[n].hist(sample)
    axs[n].set_title('n={}'.format(sample_size))

plt.show()
```


Histogram example

```
import numpy as np
import matplotlib.pyplot as plt

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2,2,
    sharex=True)
axs = [ax1, ax2, ax3, ax4]

for n in range(0, len(axs)):
    sample_size = 10**(n+1)
    sample = np.random.normal(loc=0.0, scale=1.0,
        size=sample_size)
    axs[n].hist(sample, bins=100)
    axs[n].set_title('n={}'.format(sample_size))

plt.show()
```

Histogram example



5. Scatter plot

- A kind of chart that is often used in statistics and data science
- It consists of multiple data points plotted across two axes
- Each variable depicted in a scatter plot would have various observations
- Used to identify the data's relationship with each variable (i.e., correlation, trend patterns)
- In machine learning, scatter plots are often used in regression, where x and y are continuous variable
- Also being used in clustering scatters or outlier detection

Practice with Pandas and Seaborn to manipulating data

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

iris =
pd.read_csv("../input/Iris.csv")

iris.head()
```

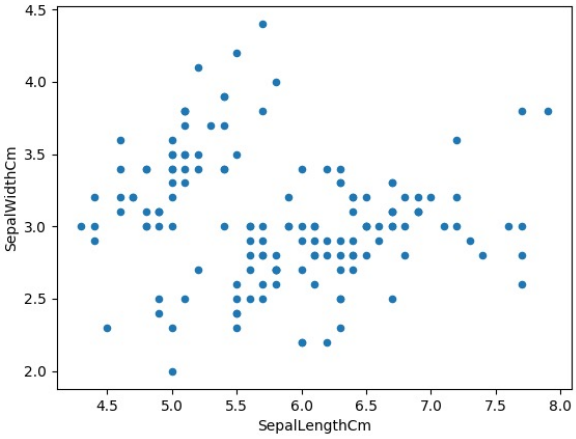
Import the dataset Iris

Practice with Pandas and Seaborn to manipulating data

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa

Use scatter plot for Iris data

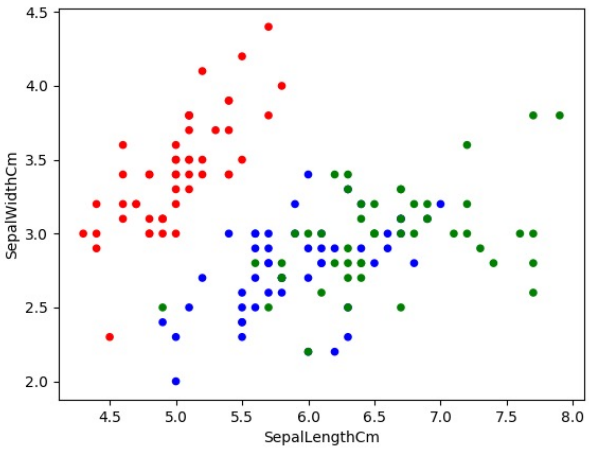
- Plot two variables: SepalLengthCm and SepalWidthCm




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

iris =
pd.read_csv("../input/Iris.csv")
iris.head()

iris["Species"].value_counts()
iris.plot(kind="scatter",
x="SepalLengthCm",
    y="SepalWidthCm")

plt.show()
```


Use scatter plot for Iris data

- Display color for each kind of Iris

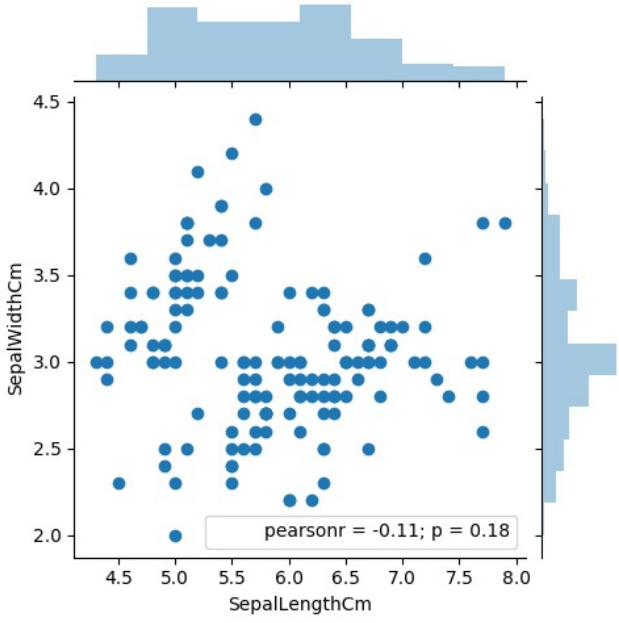



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

iris =
pd.read_csv("../input/Iris.csv")
iris.head()

iris["Species"].value_counts()
col = iris['Species'].map({"Iris-
setosa":'r',    "Iris-
virginica":'g', "Iris-
versicolor":'b'})
iris.plot(kind="scatter",
x="SepalLengthCm",
y="SepalWidthCm", c=col)

plt.show()
```

Marginal Histogram

- Histograms added to the margin of each axis of a scatter plot for analyzing the distribution of each measure
- Assess the relationship between two variables and examine their distributions

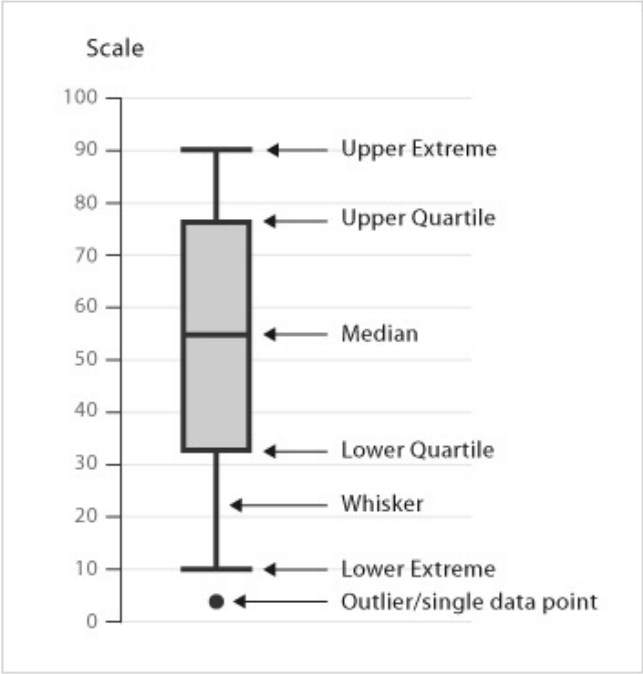
Marginal Histogram




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

iris =
pd.read_csv("../input/Iris.csv")
iris.head()

iris["Species"].value_counts()
sns.jointplot(x="SepalLengthCm",
y="SepalWidthCm",    data=iris,
size=5)

plt.show()
```

6. Other kinds of chart
Box Plot

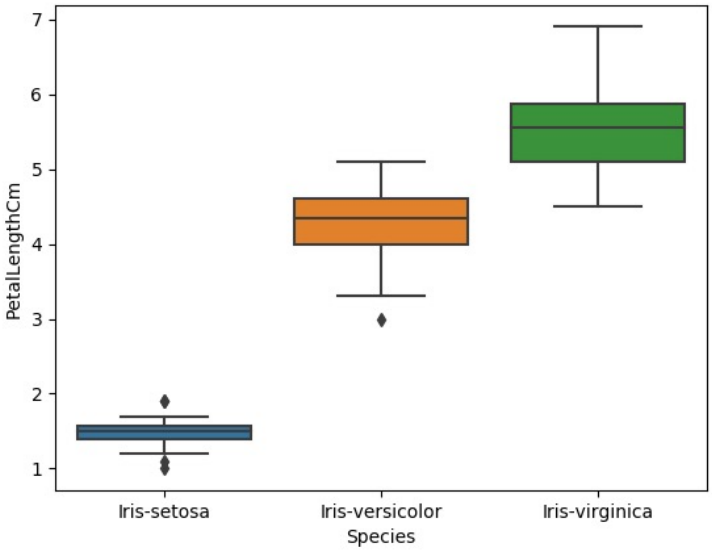
- Box and Whisker Plot (or Box Plot) is a convenient way of visually displaying the data distribution through their quartiles



Box Plot

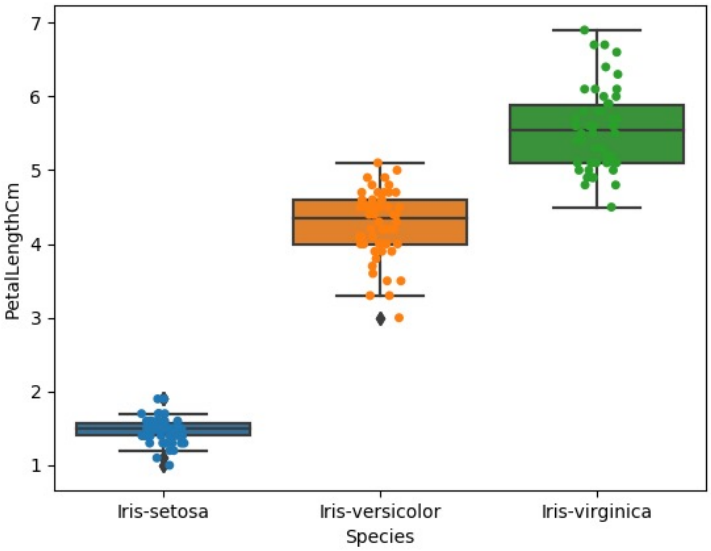
- Some observations from viewing Box Plot
  - What the key values are such as: the average, median, 25th percentile etc.
  - If there are any outliers and what their values are
  - Is the data symmetrical
  - How tightly is the data grouped
  - If the data is skewed and if so, in what direction

Box Plot




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

iris =
pd.read_csv("../input/Ir
is.csv")
iris.head()

sns.boxplot(x="Species",
    y="PetalLengthCm",
data=iris)

plt.show()
```

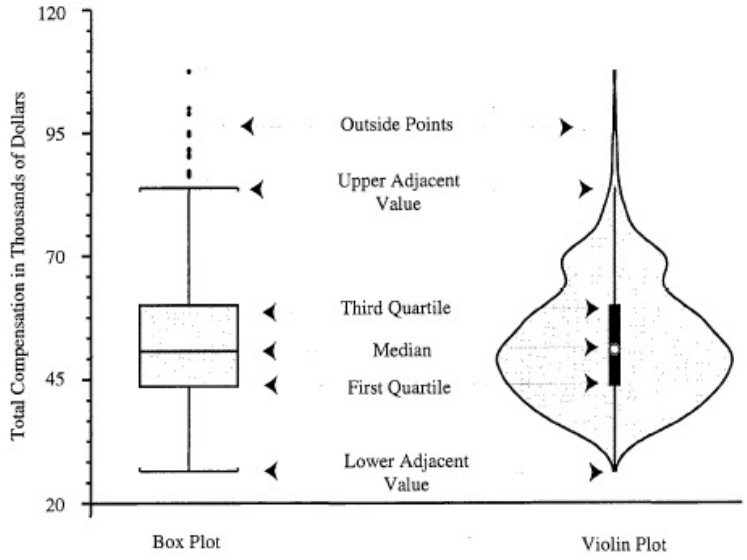

Box Plot




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

iris =
pd.read_csv("../input/Iris.
csv")
iris.head()

ax =
sns.boxplot(x="Species",
    y="PetalLengthCm",
    data=iris)
ax =
sns.stripplot(x="Species",
    y="PetalLengthCm",
data=iris,  jitter=True,
    edgecolor="gray")

plt.show()
```

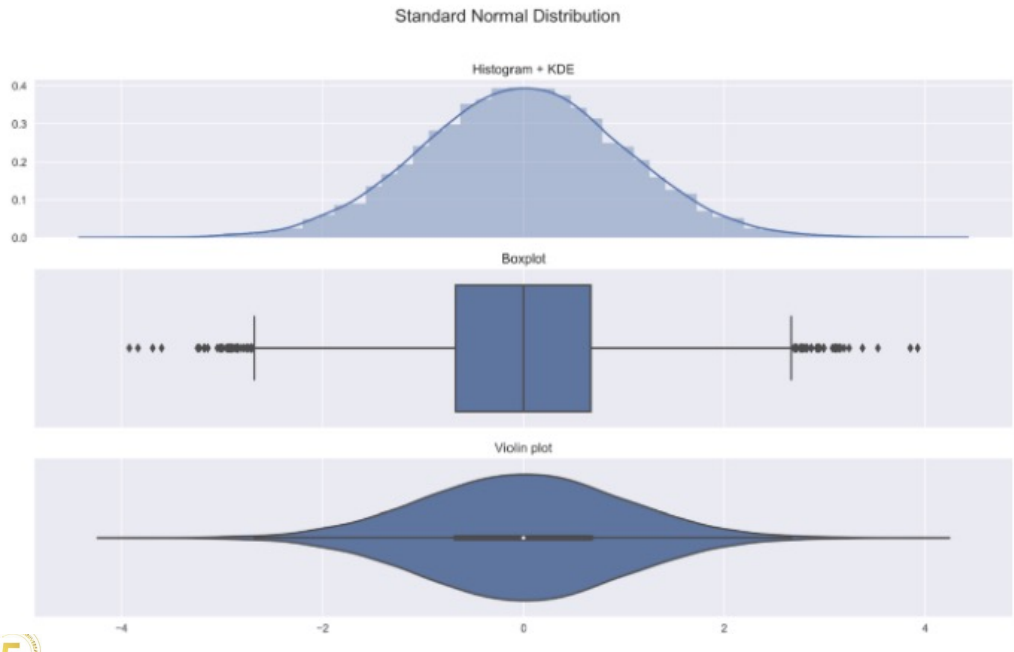
Violin Plot

- Combination of the box plot with a kernel density plot
- Same information from box plot



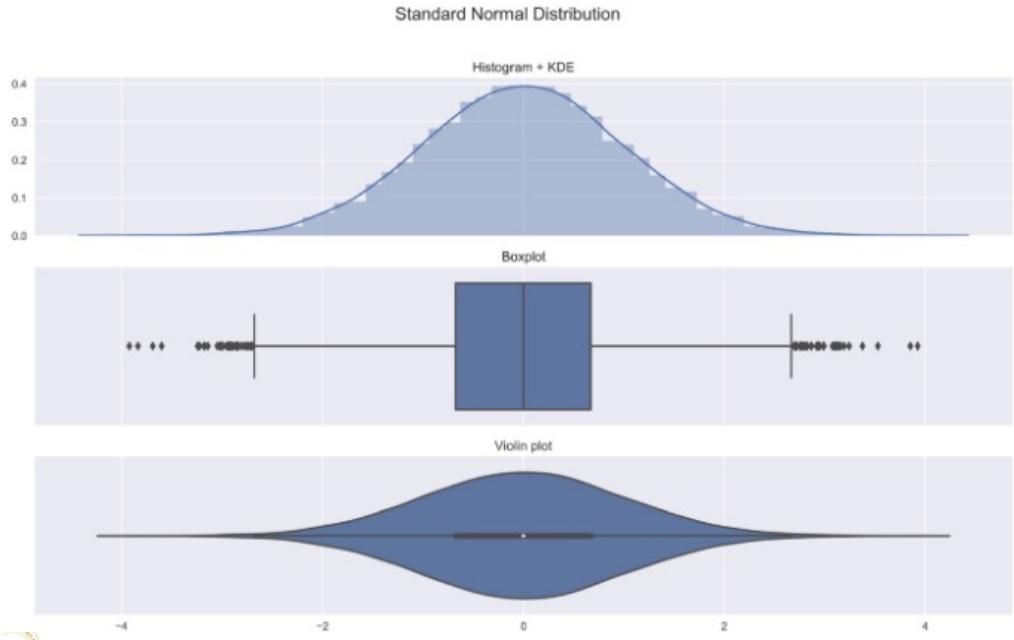
Violin Plot

- Shows the entire distribution of the data



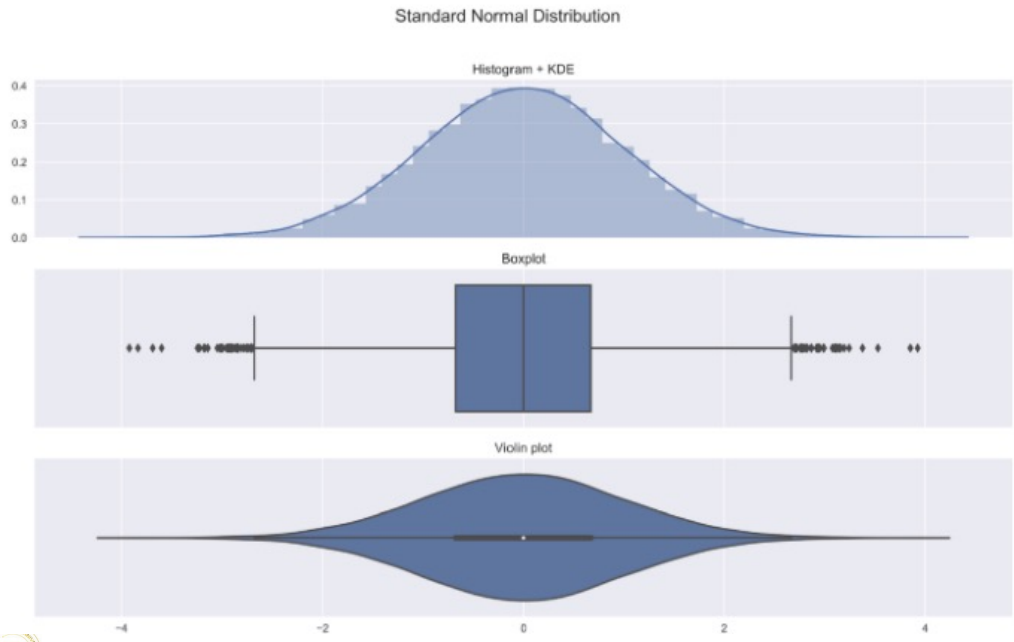
Violin Plot

- Histogram shows the symmetric shape of the distribution



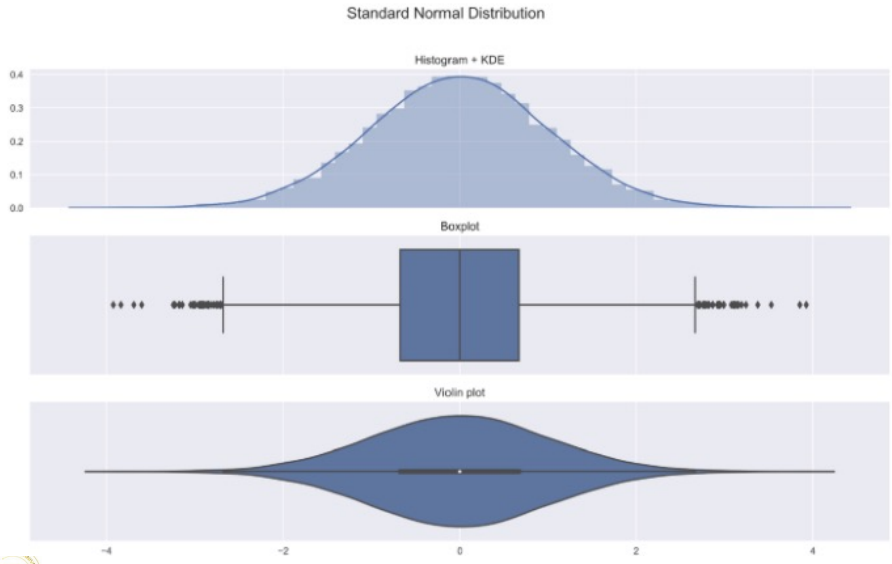
Violin Plot

- The kernel density plot used for creating the violin plot is the same as the one added on top of the histogram

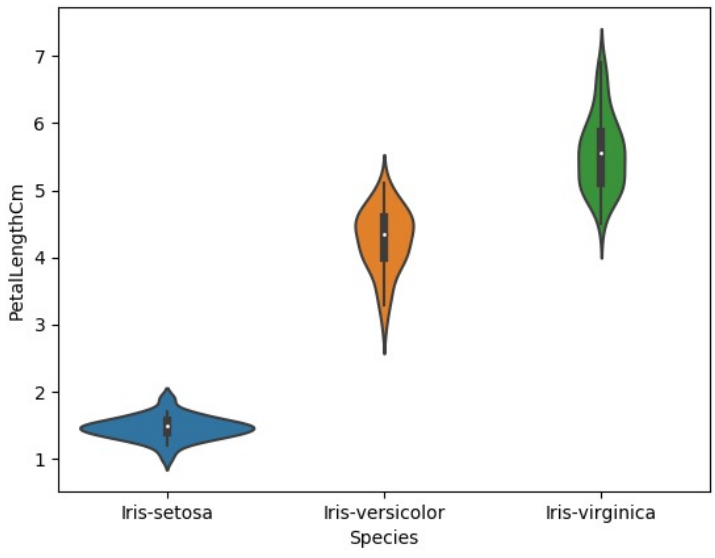


Violin Plot

- Wider sections of the violin plot represent a higher probability of observations taking a given value
- The thinner sections correspond to a lower probability.



Violin Plot of Iris data




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

iris =
pd.read_csv("../input/
Iris.csv")
iris.head()

sns.violinplot(x="Spec
ies",
y="PetalLengthCm",
data=iris, size=6)

plt.show()
```

Regression Plot

- Create a regression line between 2 parameters and helps to visualize their linear relationships
- Example: data set tips of seaborn contains information about:
  - the people who probably had food at the restaurant and whether or not they left a tip
  - the gender of the people, whether they smoke, day, time
- Use seaborn's function lmplot() to create regression plot

Regression Plot example

```
# import the library
import seaborn as sns

# load the dataset
dataset = sns.load_dataset('tips')

# the first five entries of the dataset
dataset.head()
```

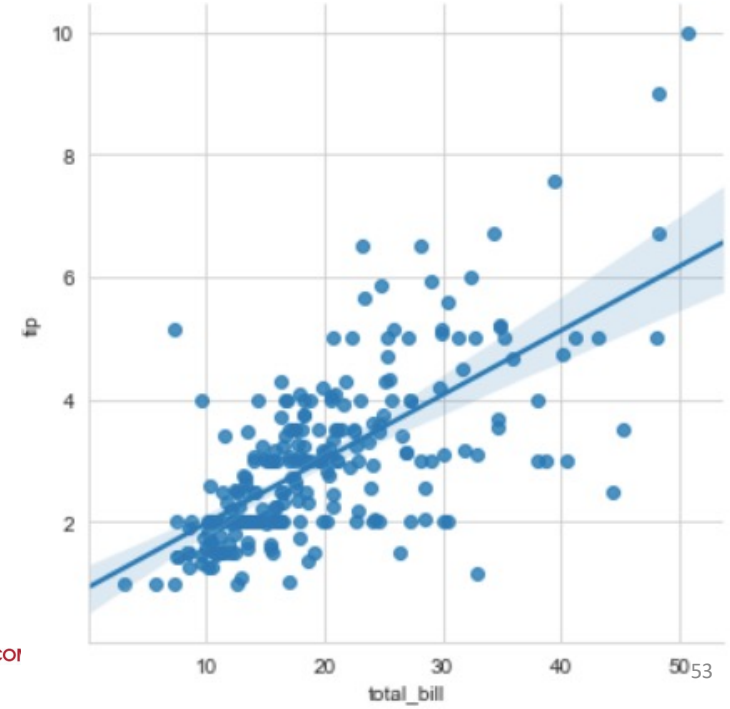
Output

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Regression Plot Example

```
sns.set_style('whitegrid')
sns.lmplot(x ='total_bill', y ='tip', data = dataset)
```

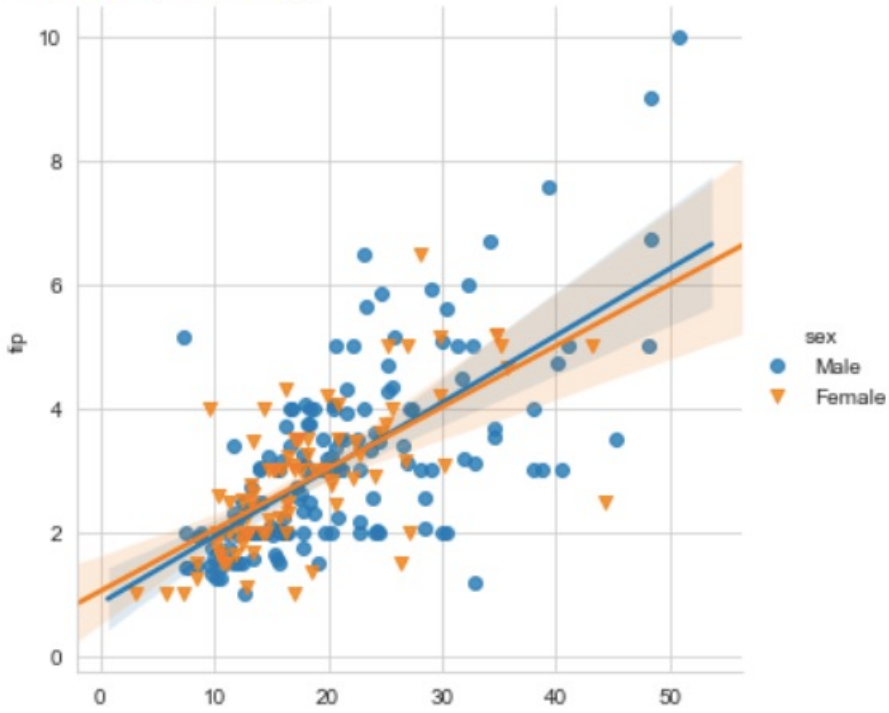
- Show the linear relationship betweet the total bill of customers and the tips they gave



Regression Plot Example

```
sns.set_style('whitegrid')
sns.lmplot(x ='total_bill', y ='tip', data = dataset,
           hue ='sex', markers =['o', 'v'])
```

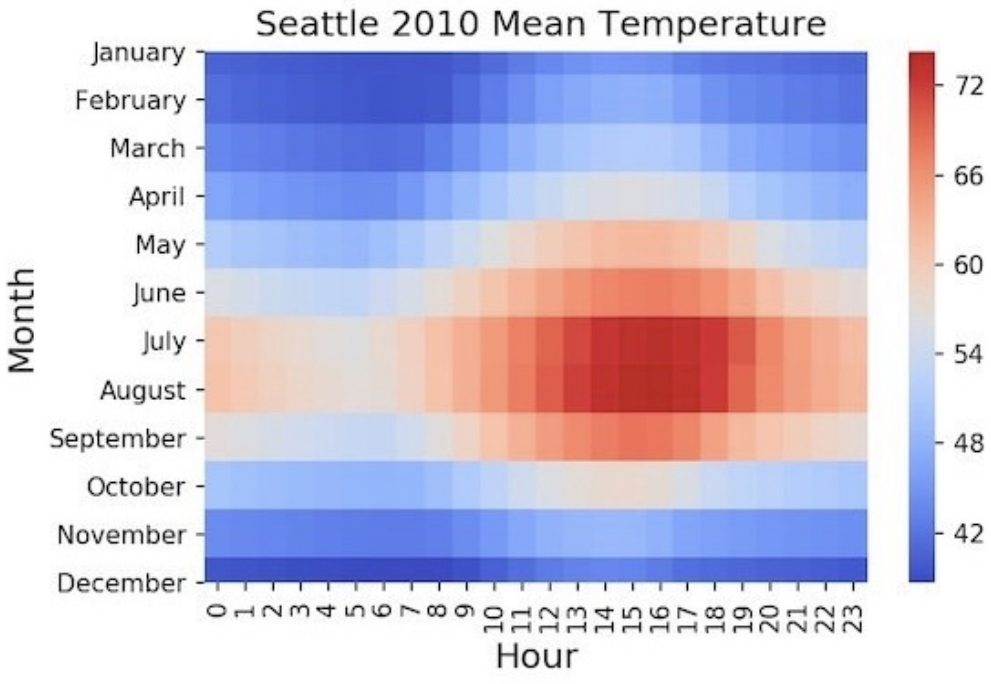
Distinguish two categories by sex



Heatmaps

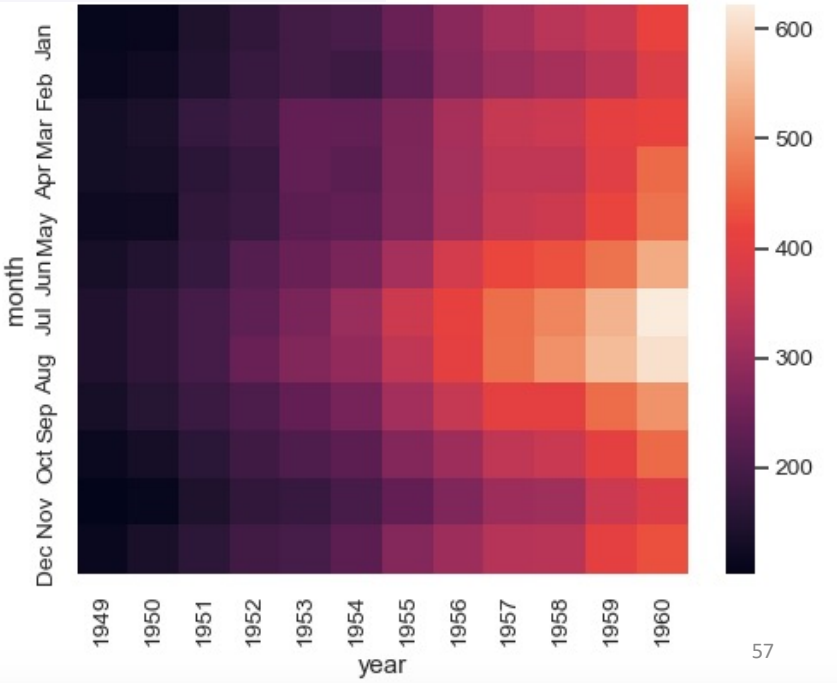
- The underlying idea: replace numbers with colors
- The goal of heatmaps is to provide a colored visual summary of information
- Heatmaps are useful for cross-examining multivariate data, through placing variables in rows and columns and coloring cells within the table
- All the rows are one category (labels displayed on the left side)
- All the columns are another category (labels displayed on the bottom)
- Data in a cell demonstrates the relationship between two variables in the connecting row and column

Heatmap Example



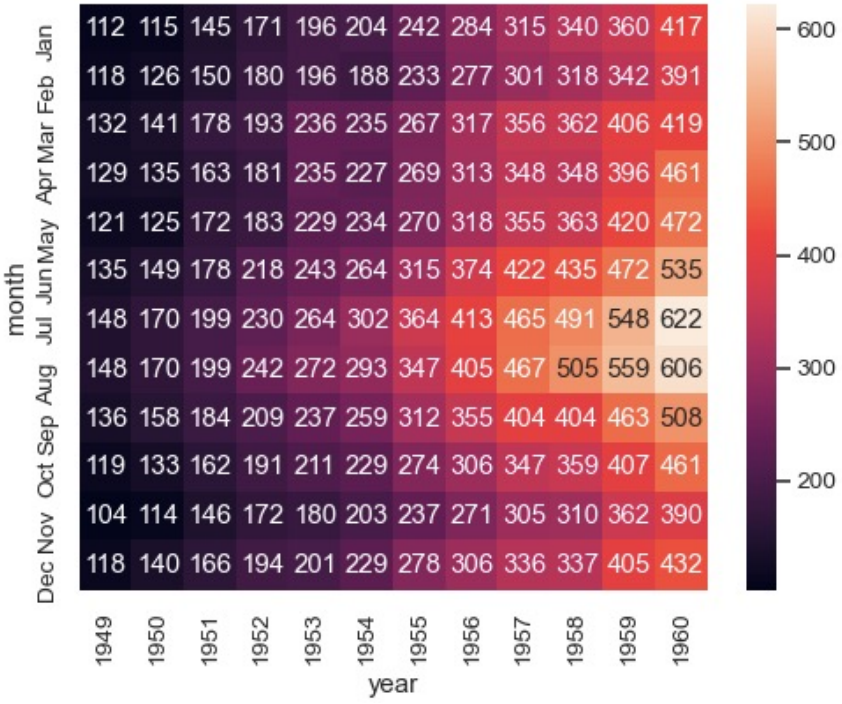
Heatmap with seaborn

```
>>> flights = sns.load_dataset("flights")
>>> flights = flights.pivot("month", "year", "passengers")
>>> ax = sns.heatmap(flights)
```

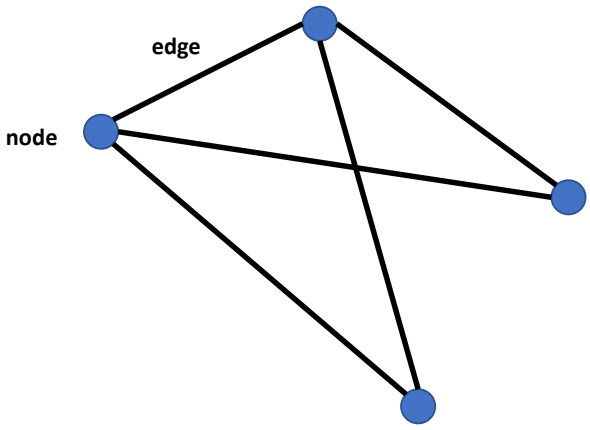


Heatmap with seaborn

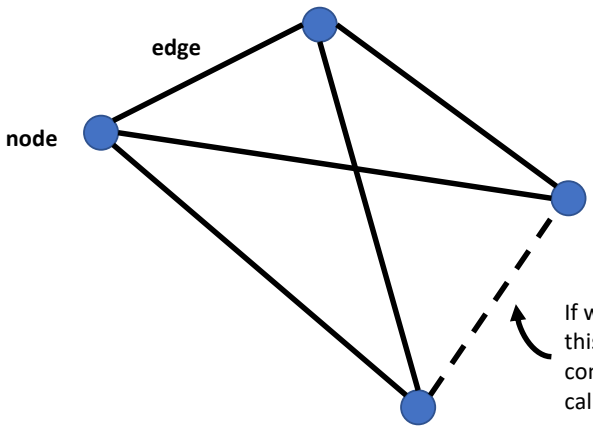
```
>>> ax = sns.heatmap(flights, annot=True, fmt="d")
```



Graphs



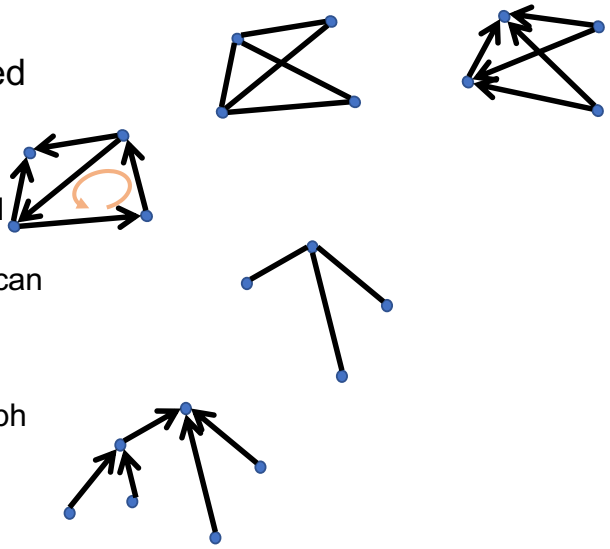
Graphs



If we add this edge then this would be a complete graph, also called a clique

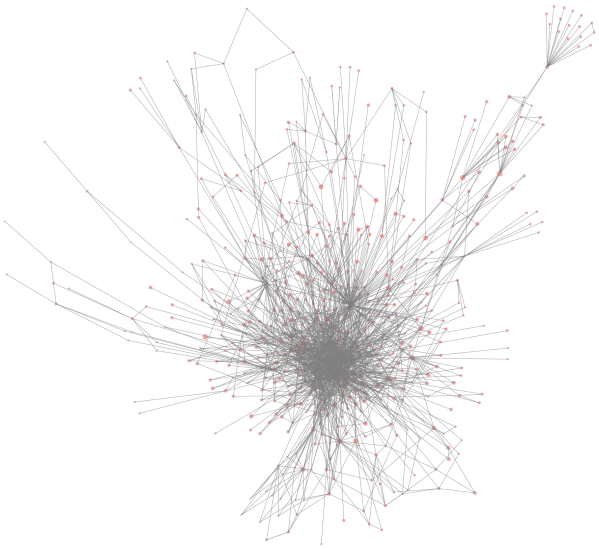
Directed Graphs and Hierarchies

- Directed vs Undirected
- Cyclic vs acyclic
- Tree
  - Minimally connected
  - N nodes, n-1 edges
  - Single parent node can have multiple child nodes
- Hierarchy
  - Acyclic directed graph
  - Having a root node



Node Degree

- Degree of a node = number of edges
- Directed graph nodes have an in-degree and an out-degree
- Social Networks
  - Many low degree nodes and fewer high degree nodes
  - Also called power-law or scale-free graphs



Graph Visualization

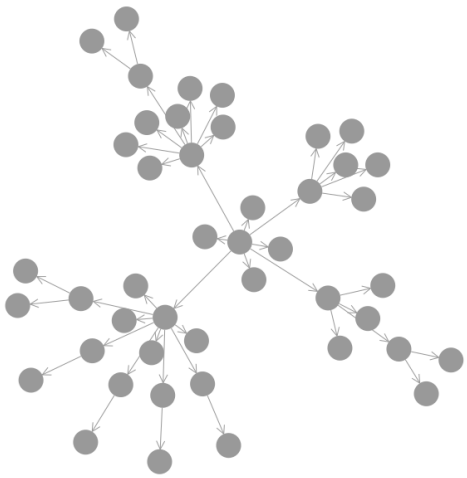
- For visualizing more abstract and non-quantitative data
- For example:
  - The relationship/contacts of individuals in a population (also called network of contacts)
  - The hierarchical structure of classes in a module
- Matplotlib does not support this kind of visualization

Roassal: an agile visualization tool

- Roassal is a DSL, written in Smalltalk and integrated in Pharo/Moose – an open source platform for software and data analysis
- Installing from: http://www.moosetechnology.org

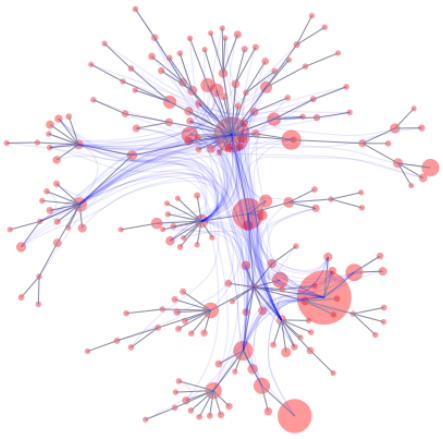
Hierarchy

```
| b |
b := RTMondrian new.
b shape circle size: 30.
b nodes: RTShape withAllSubclasses.
b shape arrowedLine
     withShorterDistanceAttachPoint
.
b edgesFrom: #superclass.
b layout forceWithCharge: -500.
b build.
^ b view
```



Network structure

```
| b lb |
b := RTMondrian new.
b shape circle color: (Color red alpha: 0.4).
b nodes: Collection withAllSubclasses.
b edges connectFrom: #superclass.
b shape
     bezierLineFollowing: #superclass;
     color: (Color blue alpha: 0.1).
b edges
     notUseInLayout;
     connectToAll: #dependentClasses.
b normalizer normalizeSize: #numberOfMethods min: 5
max: 50.
b layout force.
b build.
lb := RTLegendBuilder new.
lb view: b view.
lb addText: 'Circle = classes, size = number of
methods; gray links = inheritance;'.
lb addText: 'blue links = dependencies; layout =
force based layout on the inheritance links'.
lb build.
^ b view @ RTZoomableView
```

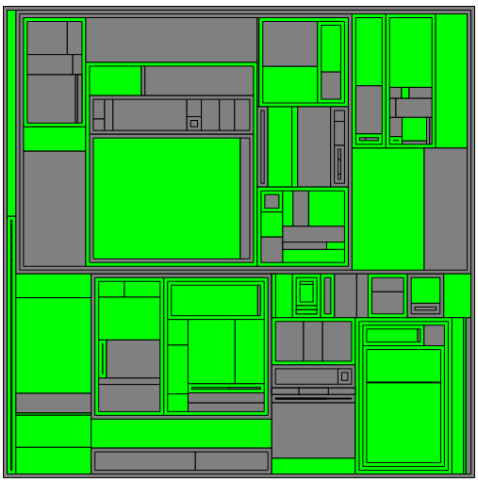


Circle = classes, size = number of methods; gray links = inheritance;

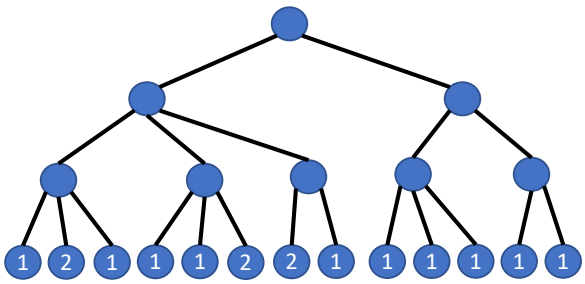
blue links = dependencies; layout = force based layout on the inheritance links

Tree Map

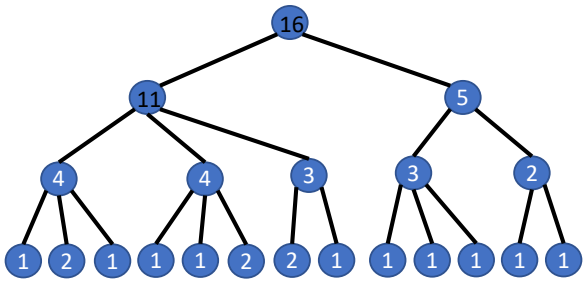
- Maps quantities to area
- Color used to differentiate areas
- Shading delineates hierarchical regions
- When to use?
  - Limited space but large amount of hierarchical data
  - Values can be aggregated in the tree structure
- Advantages
  - Saving space, display a large number of item simultaneously
  - Using color and size of areas to detect special sample data



Tree map layout

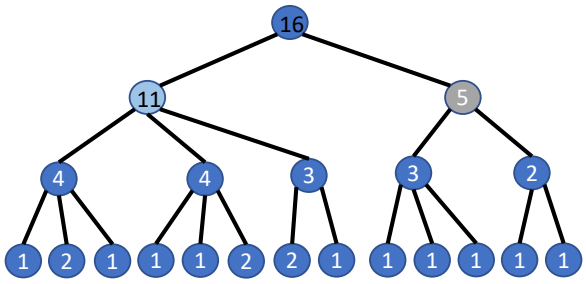


Tree map layout

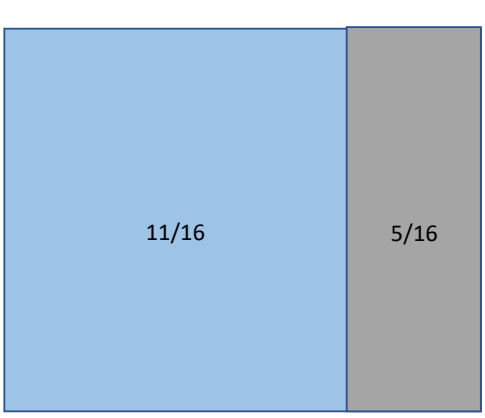


- Set parents node values to sum of child node values from bottom up

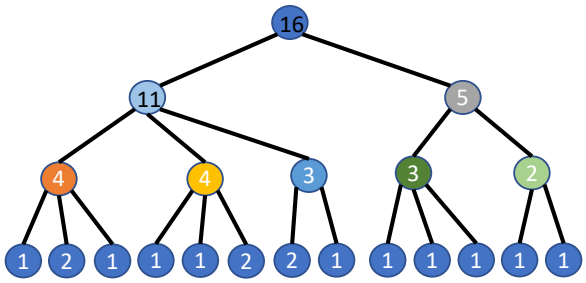
Tree map layout



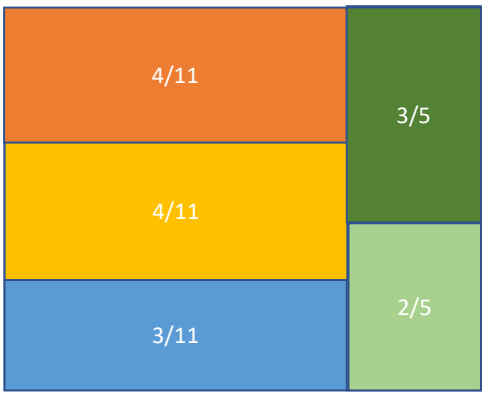
- Set parents node values to sum of child node values from bottom up
- Partition based on current node's value as a portion of parent node's value from top down



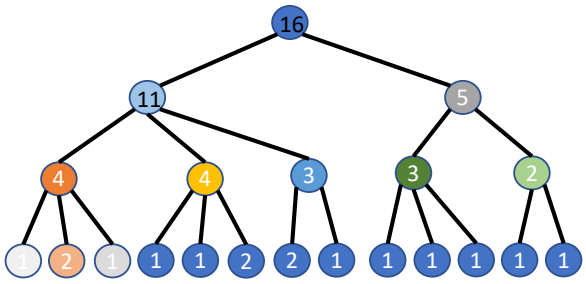
Tree map layout



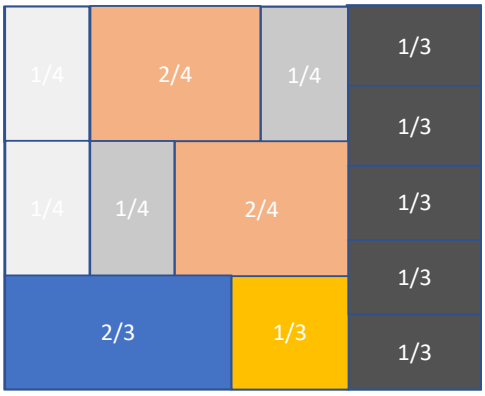
- Set parents node values to sum of child node values from bottom up
- Partition based on current node's value as a portion of parent node's value from top down



Tree map layout

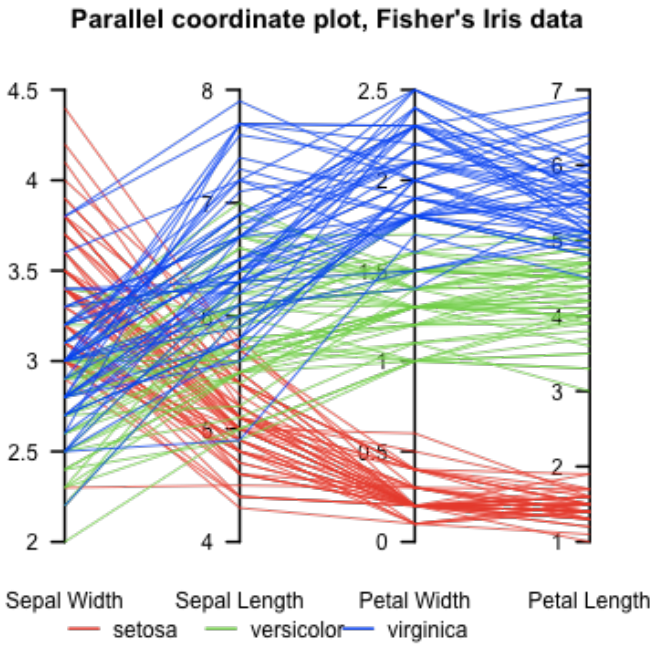


- Set parents node values to sum of child node values from bottom up
- Partition based on current node's value as a portion of parent node's value from top down



8. Multivariable Visualization

- For **data tables** with *n>3* variables: **parallel coordinates**
  - Each vertical line corresponds to a variable
  - A point in *n*-dimensional space is represented as a **polyline** with **vertices** on the parallel axes
    - the position of the vertex on the i-th axis corresponds to the value of the i-th attribute for this record
  - It might be interesting to try different axis arrangements

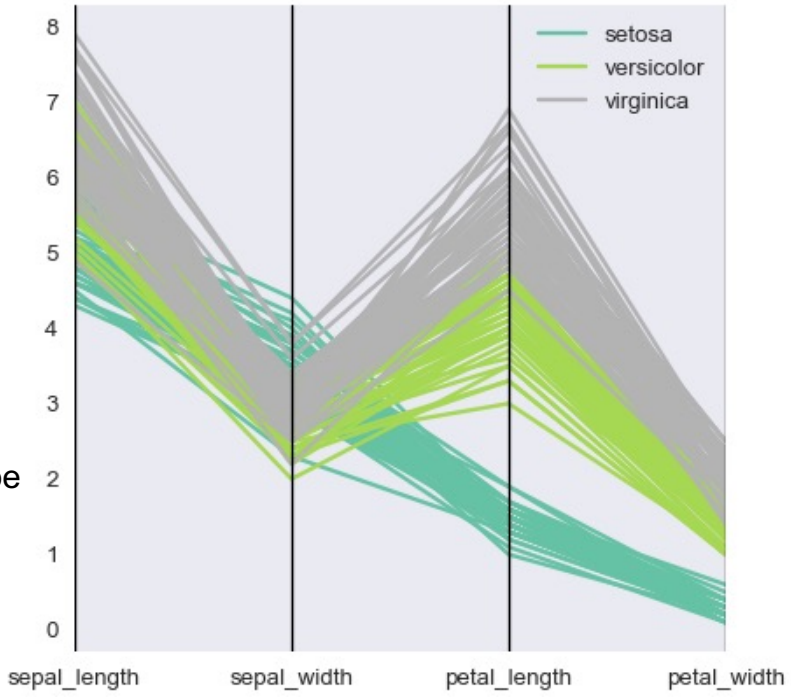


Parallel Plot

- Parallel Coordinates Plots allow to compare the feature of several individual observations on a set of numerical variables
- Each vertical bar represents a variable and usually has its own scale
- Values are plotted as series of lines connected across each axis
- Color can be used to represent different groups of individuals or highlight a specific one
- Allow to compare variations of adjacent axis
  - Changing the order can lead to the discovery of new patterns in the dataset

Parallel plot with pandans for Iris data

```
# libraries
import pandas
import matplotlib.pyplot as pl
from pandas.tools.plotting imp

# Take the iris dataset
import seaborn as sns
data = sns.load_dataset('iris'

# Make the plot
parallel_coordinates(data, 'sp
plt.show()
```



- Samples are grouped in 3 species
- Setosa seems have smaller petals but its sepal tends to be wider

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!!!

soict.hust.edu.vn/ fb.com/groups/soict