

**School of Built Environment, Engineering and Computing**

****

**Week 1: Part 1 – Graphs**

**By**

**Prof. Ah-Lian Kor**

**Graphs**

**Read in a table in csv file**

**Type the following:**

import pandas as pd

table = pd.read\_csv("cars.csv")

table.head()

**Read in a table in excel file**

**Type the following:**

import pandas as pd

table = pd.read\_excel("client.xlsx")

table.head()

A very simple introduction to data and graphs could be found here: <https://www.mathsisfun.com/data/index.html>

<https://www.w3schools.com/python/python_ml_scatterplot.asp>

**Matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python (<https://matplotlib.org/>) .

Introduction <https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py>

<https://matplotlib.org/tutorials/index.html>

[**https://scriptverse.academy/tutorials/python-matplotlib-plot-straight-line.html**](https://scriptverse.academy/tutorials/python-matplotlib-plot-straight-line.html)

**Scatter Plot**

Scatter Plot uses points (or dots) to represent the relationship between two (or more) numeric variables. It is a quick way to identify anomalies in a dataset.

**2D Plots**

Type the following codes in Jupyter Notebook. Note: The show() function causes the figure to be displayed below in[] cell without out[] with number.:

from matplotlib import pyplot as plt

height = [155, 167, 168, 177, 162,170,162,169,174,161,172,169,166]

weight = [60,86,87,88,75,86,70,87,94,78,86,85,86]

plt.scatter(height, weight)

plt.show()

**Add Title and Axes Labels to the Chart**

**Type the following (note: # is for comment):**

*#libraries*

from matplotlib import pyplot as plt

height = [155, 167, 168, 177, 162,170,162,169,174,161,172,169,166]

weight = [60,86,87,88,75,86,70,87,94,78,86,85,86]

*#Create Scatter Plot*

plt.scatter(height, weight)

*#Add title and axis names*

plt.title('A graph of Weight against Height')

plt.xlabel('Height in cm')

plt.ylabel('Weight in kg')

*# Show graphic*

plt.show()

**Plot Straight Lines**

An equation of a straight line is y = mx + c (where m is the gradient, and c is the intercept).

Type the following:

#libraries

import matplotlib.pyplot as plt

import numpy as np

#range of x values from -5 to 5 with 120 evenly spaced points over the range

x = np.linspace(-5,5,120)

#Equation of straight line

y = 3\*x+1

#Plot graph with title, axis labels, grid, and legend. m is for magenta colour and – is for continuous line

plt.plot(x, y, '-m', label='y=3x+1')

plt.title('Graph of y=3x+1')

# hex color codes can be found here https://www.color-hex.com/

plt.xlabel('x', color='#001e3d')

plt.ylabel('y', color='#001e3d')

plt.legend(loc='lower right')

plt.grid()

plt.show()

**Multiple Straight Lines**

**Definition of spines – see here** <https://matplotlib.org/3.1.0/gallery/ticks_and_spines/spines.html>

**Definition and subplots examples**

<https://matplotlib.org/3.1.0/gallery/subplots_axes_and_figures/subplots_demo.html#:~:text=pyplot.,the%20individual%20plots%20are%20created.&text=add_subplot%20for%20adding%20subplots%20at%20arbitrary%20locations%20within%20the%20figure>.

**The following codes are adapted from here:** <https://scriptverse.academy/tutorials/python-matplotlib-plot-straight-line.html>

Type the following codes:

#Libraries

import matplotlib.pyplot as plt

import numpy as np

#Plot and subplots

fig = plt.figure()

#subplot (nrows, ncolumns, index)

ax = fig.add\_subplot(1, 1, 1)

#x values range from -5 to 5 and 100 evenly spaced points between them

x = np.linspace(-5,5,100)

ax.spines['left'].set\_position('center')

#spines – see definition above

ax.spines['bottom'].set\_position('center')

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

#ticks are the markers on the axes

ax.xaxis.set\_ticks\_position('bottom')

ax.yaxis.set\_ticks\_position('left')

#plot several lines, colour and line type see here:

# <https://scriptverse.academy/tutorials/python-matplotlib-plot-straight-line.ht>

plt.plot(x, 2\*x+1, '-r', label='y=2x+1')

plt.plot(x, 2\*x-1,'-.g', label='y=2x-1')

plt.plot(x, 2\*x+3,':b', label='y=2x+3')

plt.plot(x, 2\*x-3,'--m', label='y=2x-3')

plt.plot(x, 2\*x-5, '-y', label ='y=2x-5')

plt.legend(loc='upper left')

plt.show()

**The following codes are adapted from here:** [**https://matplotlib.org/3.1.0/gallery/subplots\_axes\_and\_figures/subplots\_demo.html#:~:text=pyplot.,the%20individual%20plots%20are%20created.&text=add\_subplot%20for%20adding%20subplots%20at%20arbitrary%20locations%20within%20the%20figure**](https://matplotlib.org/3.1.0/gallery/subplots_axes_and_figures/subplots_demo.html#:~:text=pyplot.,the%20individual%20plots%20are%20created.&text=add_subplot%20for%20adding%20subplots%20at%20arbitrary%20locations%20within%20the%20figure)**.**

**Type the following codes:**

import matplotlib.pyplot as plt

import numpy as np

# Some example data to display

x = np.linspace(0, 2 \* np.pi, 400)

y = np.sin(x \*\* 2)

fig, axs = plt.subplots(2, 2)

axs[0, 0].plot(x, y)

axs[0, 0].set\_title('Axis [0,0]')

axs[0, 1].plot(x, y, 'tab:orange')

axs[0, 1].set\_title('Axis [0,1]')

axs[1, 0].plot(x, -y, 'tab:green')

axs[1, 0].set\_title('Axis [1,0]')

axs[1, 1].plot(x, -y, 'tab:red')

axs[1, 1].set\_title('Axis [1,1]')

for ax in axs.flat:

ax.set(xlabel='x-label', ylabel='y-label')

# Hide x labels and tick labels for top plots and y ticks for right plots.

for ax in axs.flat:

ax.label\_outer()

**3D Scatter Plot**

Semantics:

"111" means "1x1 grid, first subplot" and "234" means "2x3 grid, 4th subplot".

add\_subplot(111) is the same as add\_subplot(1,1,1). <https://stackoverflow.com/questions/3584805/in-matplotlib-what-does-the-argument-mean-in-fig-add-subplot111>

The following codes are adapted from here:

Type the following:

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

x =[1,2,3,4,5,6,7,8,9,10]

y =[5,6,2,3,13,4,1,2,4,8]

z =[2,3,3,3,5,7,9,11,9,10]

ax.scatter(x, y, z, c='g', marker='x')

ax.set\_xlabel('X Label')

ax.set\_ylabel('Y Label')

ax.set\_zlabel('Z Label')

plt.show()

Explore with different colours and style of markers.

Reference for colours <https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html>

Style of markers

<https://matplotlib.org/3.1.1/api/markers_api.html#module-matplotlib.markers>

**Import Data into Python**

**Read a csv file.**

Upload the cars.csv file onto the Jupyter Notebook DAV folder.

Type the following:

import pandas as pd

df = pd.read\_csv(r'cars.csv') #read the csv file (put 'r' before the path string to address any special characters in the path, such as '\'). Don't forget to put the file name at the end of the path + ".csv"

print (df)

**Read an excel file**.

Upload the client.xlsx file onto the Jupyter Notebook DAV folder.

Type the following:

import pandas as pd

df = pd.read\_excel(r'client.xlsx') #read the excel file (put 'r' before the path string to address any special characters in the path, such as '\'). Don't forget to put the file name at the end of the path + ".csv"

print (df)

To display selected columns, type the following:

import pandas as pd

data = pd.read\_excel (r'client.xlsx')

df = pd.DataFrame(data, columns= ['Client\_Name','Country'])

print (df)

**Plot from a text file (text1.txt with data format: int1, int2)**

[**http://www.learningaboutelectronics.com/Articles/How-to-plot-a-graph-with-matplotlib-from-data-from-a-CSV-file-using-the-CSV-module-in-Python.php**](http://www.learningaboutelectronics.com/Articles/How-to-plot-a-graph-with-matplotlib-from-data-from-a-CSV-file-using-the-CSV-module-in-Python.php)

**Type 1: Text File**

import pandas as pd

df = pd.read\_csv("text1.txt",delimiter=',')

plt.plot(x,y, marker='o')

plt.title('Data from the CSV File: People and Expenses')

plt.xlabel('Number of People')

plt.ylabel('Expenses')

plt.show()

**Type 2: Excel File with Date**

**Note: you have to parse the date first**

**Type the following:**

import pandas as pd

df = pd.read\_excel('date\_data.xlsx', parse\_dates=True,index\_col=0,

names = ['Date','Temperature'])

df.plot()

**Type 3: Plot with Date**

**Type the following codes with data:**

import pandas as pd

import matplotlib.pyplot as plt

date\_time = ["2011-09-01", "2011-08-01", "2011-07-01", "2011-06-01", "2011-05-01"]

date\_time = pd.to\_datetime(date\_time)

temp = [2, 4, 6, 4, 6]

DF = pd.DataFrame()

DF['temp'] = temp

DF = DF.set\_index(date\_time)

fig, ax = plt.subplots()

fig.subplots\_adjust(bottom=0.3)

plt.xticks(rotation=90)

plt.plot(DF)

**Type the following codes with data from a file:**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("temperature.txt",delimiter=',', names = ['Date', 'Temperature'])

df['Date'] = pd.to\_datetime(df["Date"])

df = df.set\_index('Date')

fig, ax = plt.subplots()

ax.plot(df.index, df['Temperature'])

plt.title('Temperature Time Series')

plt.xlabel('Date')

plt.ylabel('Temperature in Degrees Fahrenheit')

plt.xticks(rotation=90)

**Type the following codes with data from a file:**

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_excel('date\_data.xlsx',header=None,names=['time','temperature'])

data.time = pd.to\_datetime(data['time'], format='%Y-%m-%d %H:%M:%S.%f')

data.set\_index(['time'],inplace=True)

data.plot()

**Resources:**

<https://pythonprogramming.net/loading-file-data-matplotlib-tutorial/>

<http://www.learningaboutelectronics.com/Articles/How-to-plot-a-graph-with-matplotlib-from-data-from-a-CSV-file-using-the-CSV-module-in-Python.php>

<https://plotly.com/python/plot-data-from-csv/>

<https://www.datacamp.com/community/tutorials/python-excel-tutorial>

<https://datatofish.com/import-csv-file-python-using-pandas/>

<https://datatofish.com/read_excel/>

<https://www.codegrepper.com/code-examples/delphi/how+to+read+excel+file+in+jupyter+notebook>

**How to save charts as:**

**Do the following exercises on your own:**

**PDF** [**https://datatofish.com/export-matplotlib-pdf/**](https://datatofish.com/export-matplotlib-pdf/)

**Image** [**https://chartio.com/resources/tutorials/how-to-save-a-plot-to-a-file-using-matplotlib/**](https://chartio.com/resources/tutorials/how-to-save-a-plot-to-a-file-using-matplotlib/)

**Pie Chart**

**Example 1 – Exploded Pie Chart with Shadow (Use Matplotlib)**

**Type the following:**

import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'

sizes = [15, 30, 45, 10]

explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

shadow=True, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

**Reference**

<https://matplotlib.org/3.1.1/gallery/pie_and_polar_charts/pie_features.html>

**Example 2 – Use Panda Data Frame**

**Type the following:**

import pandas as pd

df = pd.DataFrame({'mass': [0.330, 4.87 , 5.97],

'radius': [2439.7, 6051.8, 6378.1]},

index=['Mercury', 'Venus', 'Earth'])

plot = df.plot.pie(y='mass', figsize=(5, 5))

Note: explore with different figure sizes and also plot a pie chart for the radius.

**Resources**

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html>

<https://stackoverflow.com/questions/56294597/how-to-create-a-pie-chart-from-csv-file-using-python>

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.pie.html>

<https://matplotlib.org/3.1.1/gallery/pie_and_polar_charts/pie_features.html>

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.pie.html>

**Pie Chart: Read Data from File and Plot the Pie Chart**

**Type the following codes:**

import pandas as pd

from matplotlib.pyplot import pie, axis, show

%matplotlib inline

df = pd.read\_excel ('client.xlsx')

sums = df.groupby(df["Product"])["Price"].sum()

axis('equal');

pie(sums, labels=sums.index);

show()

**Resources:**

[**https://pbpython.com/simple-graphing-pandas.html**](https://pbpython.com/simple-graphing-pandas.html)

**Bar Chart**

**Simple Bar Chart**

**Type the following:**

#simple bar chart

import pandas as pd

df = pd.DataFrame({'categories':['A', 'B', 'C'], 'value':[10, 30, 20]})

ax = df.plot.bar(x='categories', y='value', color = "r", rot=0) # rot is rotation for the labels, could be rot = 90

#set the x-axis label

ax.set\_xlabel("Categories")

#set the y-axis label

ax.set\_ylabel("Frequency")

**Resources**

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html)

**Two-Bar Chart**

**Resources Used**

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html)

**Type 1: Side by Side**

**Type the following codes:**

#2-bar chart

import pandas as pd

speed = [0.1, 17.5, 40, 48, 52, 69, 88]

lifespan = [2, 8, 70, 1.5, 25, 12, 28]

index = ['snail', 'pig', 'elephant', 'rabbit', 'giraffe', 'coyote', 'horse']

df = pd.DataFrame({'speed': speed,

'lifespan': lifespan}, index=index)

ax = df.plot.bar(color = ("g", "m"), width= 0.85, rot=90)

# set title

ax.set\_title('Speed and Lifespan of Animals')

#set the x-axis label

ax.set\_xlabel("Animals")

#set the y-axis label

ax.set\_ylabel("Units")

**Type 2: Stacked**

**Type the following:**

#Stacked bar chart

import pandas as pd

speed = [0.1, 17.5, 40, 48, 52, 69, 88]

lifespan = [2, 8, 70, 1.5, 25, 12, 28]

index = ['snail', 'pig', 'elephant', 'rabbit', 'giraffe', 'coyote', 'horse']

df = pd.DataFrame({'speed': speed,

'lifespan': lifespan}, index=index)

ax = df.plot.bar(stacked=True, color = ("b", "k"), width= 0.6, rot = 0)

# set title

ax.set\_title('Speed and Lifespan of Animals')

#set the x-axis label

ax.set\_xlabel("Animals")

#set the y-axis label

ax.set\_ylabel("Units")

**Type 3: Subplots**

**Type the following:**

import pandas as pd

speed = [0.1, 17.5, 40, 48, 52, 69, 88]

lifespan = [2, 8, 70, 1.5, 25, 12, 28]

index = ['snail', 'pig', 'elephant', 'rabbit', 'giraffe', 'coyote', 'horse']

df = pd.DataFrame({'speed': speed,

'lifespan': lifespan}, index=index)

# color hexa codes https://www.w3schools.com/colors/colors\_picker.asp

axes = df.plot.bar(rot=0, color = ("#ffbf00", "#bfff00" ), subplots=True)

axes[1].legend(loc=2)

**Resources**

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html)

[**https://matplotlib.org/3.3.2/api/\_as\_gen/matplotlib.pyplot.bar.html**](https://matplotlib.org/3.3.2/api/_as_gen/matplotlib.pyplot.bar.html)

**Type 4: Multiple Plots**

**Type the following:**

#https://matplotlib.org/3.3.2/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py

import matplotlib.pyplot as plt

names = ['group\_a', 'group\_b', 'group\_c']

values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)

plt.bar(names, values)

plt.subplot(132)

plt.scatter(names, values)

plt.subplot(133)

plt.plot(names, values)

plt.suptitle('Categorical Plotting')

plt.show()

**Type 5: Read Data from File**

We are going to read the sales.txt file. Next we are going to plot some bar charts from the data.

**Part 1: display the entire sales.txt dataset**

**Type the following:**

#Plot some data

# Carry on from here https://pbpython.com/simple-graphing-pandas.html

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

sales.head()

**Part 2: display the relevant columns of the sales.txt dataset**

**Type the following in a new cell:**

#Plot some data

# Carry on from here https://pbpython.com/simple-graphing-pandas.html

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

#We are only interested in 3 columns of the dataset

customers = sales[['name','ext price','date']]

customers.head()

**Part 3: We would like to group the data by customer names so that we know the purchase distribution by customer**

**Type the following in a new cell:**

#Plot some data

# Carry on from here https://pbpython.com/simple-graphing-pandas.html

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

#We are only interested in 3 columns of the dataset

customers = sales[['name','ext price','date']]

customer\_group = customers.groupby('name')

customer\_group.size()

**Part 4: We would like to know the sales total for each customer and sort the data based on the column ‘ext price’ aggregated by customer. We are going to use pandas and dataframe.**

**Type the following in a new cell:**

#Plot some data

# Carry on from here https://pbpython.com/simple-graphing-pandas.html

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

#We are only interested in 3 columns of the dataset

customers = sales[['name','ext price','date']]

customer\_group = customers.groupby('name')

sales\_totals = customer\_group.sum()

df = pd.DataFrame(data = sales\_totals)

result = df.sort\_values(by=['ext price'], ascending = True)

result.head()

**Part 5: Plot a bar chart based on Part 4**

**Type the following codes:**

#Plot some data

# Carry on from here https://pbpython.com/simple-graphing-pandas.html

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

#We are only interested in 3 columns of the dataset

customers = sales[['name','ext price','date']]

customer\_group = customers.groupby('name')

sales\_totals = customer\_group.sum()

df = pd.DataFrame(data = sales\_totals)

result = df.sort\_values(by=['ext price'], ascending = True)

myplot = result.plot(kind='bar', rot=90, color = "yellow", title = "A Graph of Total Sales Against Customers")

myplot.set\_xlabel("Names of Customers")

myplot.set\_ylabel("Total Sales in £")

**Resources:**

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html)

[**https://matplotlib.org/3.1.1/api/\_as\_gen/matplotlib.pyplot.bar.html**](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.bar.html)

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.bar.html)

**Histogram**

**Work through these on your own**

**Resources**

[**https://matplotlib.org/3.3.1/api/\_as\_gen/matplotlib.pyplot.hist.html**](https://matplotlib.org/3.3.1/api/_as_gen/matplotlib.pyplot.hist.html)

[**https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.hist.html)

**Time Series Graphs**

**Read Data from sales.txt File**

**Note resample is used** to **resample time**-**series** data using group by: M – month, Y – year, W - week.

**Type the following:**

# Standard import for pandas, numpy and matplot

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Read in the csv file and display some of the basic info

sales=pd.read\_csv("sales.txt",parse\_dates=['date'])

print ("Data types in the file:\n")

print (sales.dtypes)

print("\n")

print ("Summary of the input file:")

print (sales.describe())

print("\n")

print ("Basic unit price stats:")

print (sales['unit price'].describe())

print("\n")

#Graphing

purchase\_patterns = sales[['ext price','date']]

# Create a line chart showing purchases by month

purchase\_patterns = purchase\_patterns.set\_index('date')

month\_plot = purchase\_patterns.resample('M').sum().plot(title="Total Sales by Month",legend=None)

fig\_m = month\_plot.get\_figure()

#Show the image, then save it

plt.show()

fig\_m.savefig("total-sales-month1-1.png")

# Create a line chart showing purchases by week

week\_plot2 = purchase\_patterns.resample('W').sum().plot(title="Total Sales by Week",legend=None)

fig\_w2 = week\_plot2.get\_figure()

#Show the image, then save it

plt.show()

fig\_w2.savefig("total-sales-week1-2.png")

# Create a line chart showing mean purchases by week

week\_plot3 = purchase\_patterns.resample('W').mean().plot(title="Mean Sales by Week",legend=None)

fig\_w3 = week\_plot3.get\_figure()

#Show the image, then save it

plt.show()

fig\_w3.savefig("mean-sales-week1-3.png")

**Resources:**

[**https://pandas.pydata.org/docs/getting\_started/intro\_tutorials/04\_plotting.html**](https://pandas.pydata.org/docs/getting_started/intro_tutorials/04_plotting.html)