6. Advanced Visualisation Tools

DataKirk

Talk overview

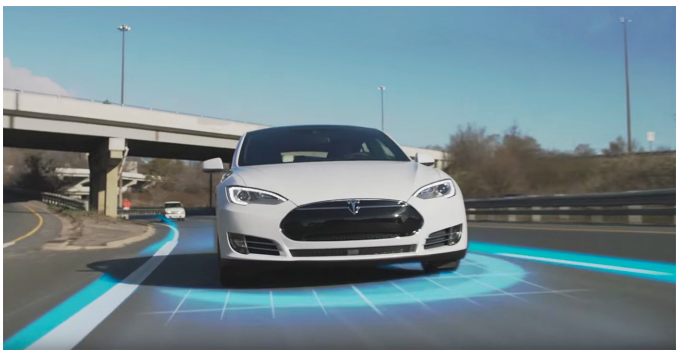
1. Example application of the week
2. Recap of last time
3. More visualisation

➡ Bar charts

➡ Vector properties

➡ Images and heat maps

Example application: self driving cars



Recap: Matplotlib

- Matplotlib is a python library - that means it's external code that someone else has written, which can be imported and used within our own code.
- The most common way to import Matplotlib is to type

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

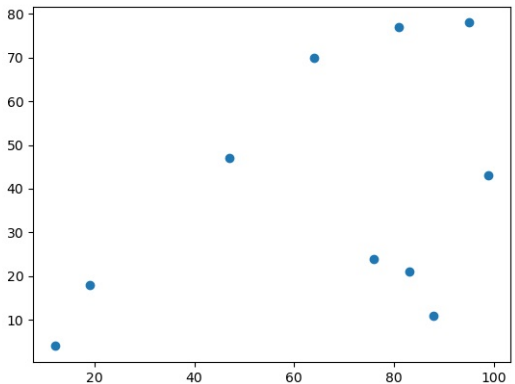
- Matplotlib contains functions that can be used for plotting. These will now be accessible by writing `plt.function_name()`.
- Each function does something slightly different, but all of them can be used to make and adjust data visualisations of different kinds.

Recap: scatter plots

- One of the most useful plot types is the scatter plot which can be accessed via `plt.scatter()`. This creates a cloud of points.
- You need to give this function two arguments:
  1. The x-coordinates of the points
  2. The y-coordinates of the points

```
# create a new figure
plt.figure()

# create x and y coordinates via lists
x = [99, 19, 88, 12, 95, 47, 81, 64, 83, 76]
y = [43, 18, 11,  4, 78, 47, 77, 70, 21, 24]

# scatter the points onto the figure
plt.scatter(x, y)
```

Recap: scatter plots

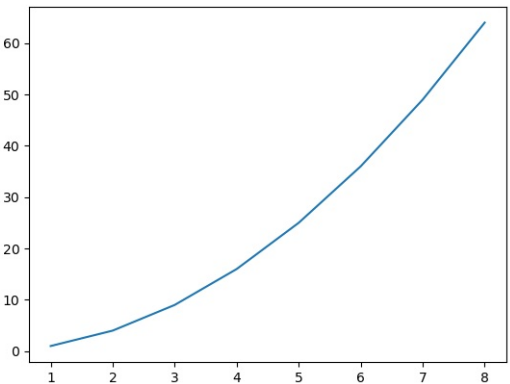


Recap: line-plots

- This is very similar to scatter, but it creates a line or curve on the graph by "connecting the dots".
- The required arguments are a sequence of x-values, and a sequence of y-values. The x-values should be in increasing order.

```
# create a new figure
plt.figure()

# create x and y values via lists
x = [1, 2, 3, 4, 5, 6, 7, 8]
y = [1, 4, 9, 16, 25, 36, 49, 64]

# plot the line
plt.plot(x, y)
```

Recap: line plots

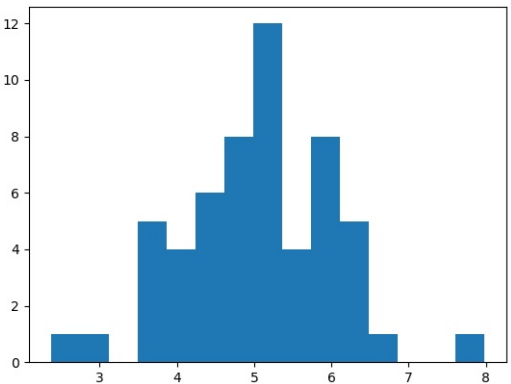


Recap: histograms

- Matplotlib can also be used to create histograms with `plt.hist()`. This function is different in that it has only one necessary argument: just a list of values (rather than a set of x points and y points as previously).

```
# create a new figure
plt.figure()

# create a list of observations
observations = [5.24, 3.82, 3.73, 5.3 , 3.93, 5.32, 6.43, 4.4 , 5.79,
4.05, 5.34, 5.62, 6.02, 6.08, 6.39, 5.03, 5.34, 4.98, 3.84, 4.91, 6.62,
4.66, 5.06, 2.37, 5.  , 3.7 , 5.22, 5.86, 3.88, 4.68, 4.88, 5.01, 3.09,
5.38, 4.78, 6.26, 6.29, 5.77, 4.33, 5.96, 4.74, 4.54, 7.99, 5.  , 4.85,
5.68, 3.73, 4.42, 4.99, 4.47, 6.06, 5.88, 4.56, 5.37, 6.39, 4.15]

# create a histogram with 15 intervals
plt.hist(observations, bins=15)
```

Recap: histograms



Recap: other plotting options

- With each plotting function, there are a number of settings you can adjust to tweak the look and feel of the graph. This is done by providing extra arguments such as `color`, `alpha`, and `label` (which needs to be used with `plt.legend()`). For example

```
# create a new figure
plt.figure()

# plot a red line with a transparancy of 40%. Label this 'line 1'
plt.plot(x, y, color='red', alpha=0.4, label='line 1')

# make a key appear on the plot
plt.legend()
```

Recap: other plotting options

- On top of this, there are options for the plot as a whole such as `plt.legend()`, `plt.xlabel()`, `plt.ylabel()`, `plt.title()`, `plt.xlim()`, `plt.ylim()` and more.

```
plt.xlabel('a great x axis')     # add a labal to the x axis
plt.ylabel('an amazing y axis')  # add a label to the y axis
plt.title('A fantastic plot')    # add a title
plt.xlim(-5, 5)                  # change the x-limits to x1=-5, x2=5
plt.ylim(0, 50)                  # change the y-limits to y1=0, y2=50
plt.yscale('log')                # make the y axis log-scale
plt.text(1, 1, 'Hello!')         # add a text annotation at x=1, y=1
# and more....
```

- Generally, if there's some feature of your plot that you would like to add, just try Googling it. There's a good chance it's possible with the right commands.

Recap: plotting with Pandas

- A common way to create Matplotlib visualisations is to use data from Pandas dataframes.
- Pandas dataframes are like tables. They have columns and rows.

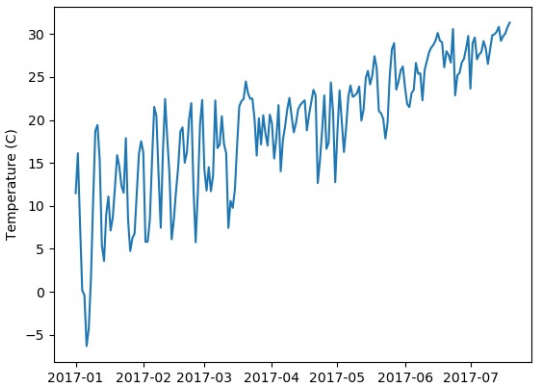
Date	Rainfall	Temp (C)
2017-01-010	0	8
2017-01-02	41	7
2017-01-03	36	9

- A single column can be selected as `datframe['Rainfall (mm)']`

Recap: plotting with Pandas

```
# import pandas
import pandas as pd

# read in data from a csv
data = pd.read_csv('path/to/file.csv', parse_dates=True)

# create a new matplotlib figure
plt.figure()

# plot the temperature over time
plt.plot(data['Date'], data['Temp (C)'])

# add a ylabel
plt.ylabel('Temperature (C)')
```

Recap: plotting with Pandas

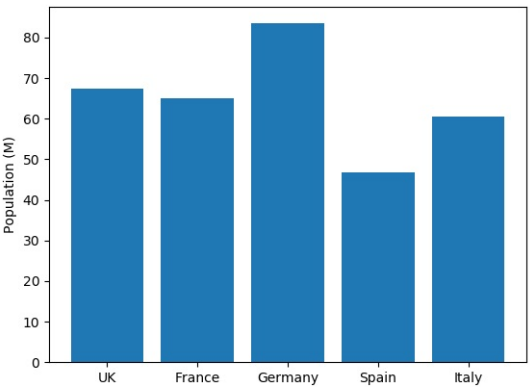


New topic: bar charts

- Bar charts are sometimes useful when you want to compare a single quantity across multiple groups. In matplotlib you can create bar charts with `plt.bar()`.
- This needs two inputs:
  1. A list of groups
  2. The height of each bar

```
plt.figure()

# create inputs
x = ['UK', 'France', 'Germany', 'Spain', 'Italy']
y = [67.5, 65.1, 83.5, 46.7, 60.6]

# plot the chart
plt.bar(x, y)
plt.ylabel('Population (M)')
```

Bar charts

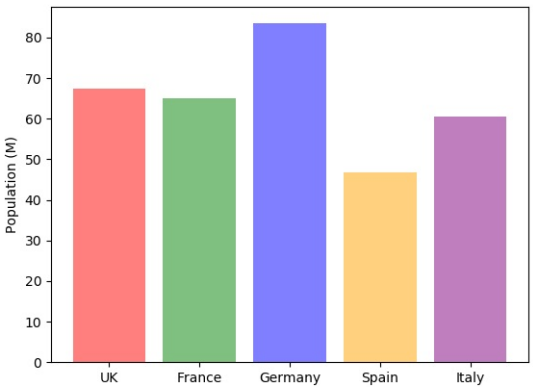

Population (M)
0
10
20
30
40
50
60
70
80
UK
France
Germany
Spain
Italy

Bar charts

- As with other plots, you can add optional arguments such as colour and transparency

```
plt.figure()

# create inputs
x = ['UK', 'France', 'Germany', 'Spain', 'Italy']
y = [67.5, 65.1, 83.5, 46.7, 60.6]

# create a list of colours
colour = ['red', 'green', 'blue', 'orange', 'purple']

# plot the chart with the colors and transparancy
plt.bar(x, y, color=colour, alpha=0.5)
plt.ylabel('Population (M)')
```

Bar charts

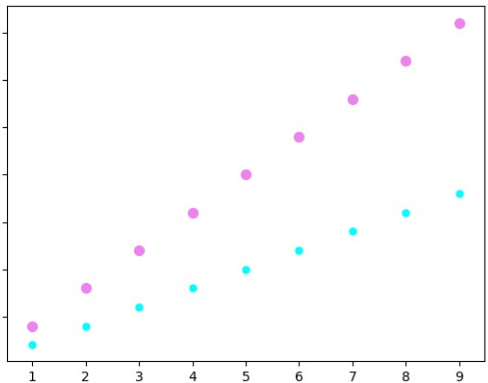

Population (M)
0
10
20
30
40
50
60
70
80
UK
France
Germany
Spain
Italy

New topic: vector properties

- In the plots covered previously, there was often the option for extra arguments in the functions such as `color`, `alpha` and `s` (size).
- We often set a single value for these, which is applied to every object in that series.

```
plt.figure()

x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y1 = [2, 4, 6, 8, 10, 12, 14, 16, 18]
y2 = [4, 8, 12, 16, 20, 24, 28, 32, 36]

plt.scatter(x, y1, color='cyan', s=5)
plt.scatter(x, y2, color='violet', s=15)
```

Vector properties

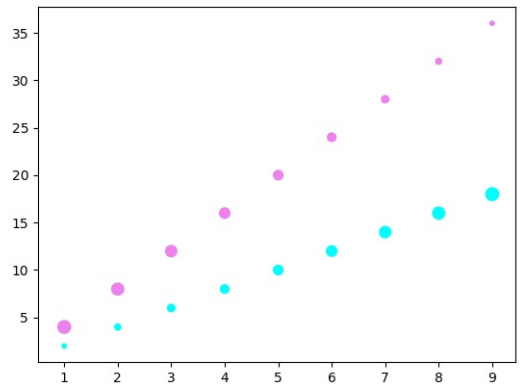


Vector properties

- However some plots, such as `plt.scatter()` allow for the properties of each individual point to be specified as a list or *vector*. This is also what happened in the second bar chart plot, when instead of giving a single colour, we passed a list of colours.

```
plt.figure()

x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y1 = [2, 4, 6, 8, 10, 12, 14, 16, 18]
y2 = [4, 8, 12, 16, 20, 24, 28, 32, 36]
size1 = [10, 20, 30, 40, 50, 60, 70, 80, 90]
size2 = [90, 80, 70, 60, 50, 40, 30, 20, 10]

plt.scatter(x, y1, color='cyan', s=size1)
plt.scatter(x, y2, color='violet', s=size2)
```

Vector properties



Vector properties: example

- In this example we will create a scatter plot where each point represents a country. The x-axis will show the country's GDP per capita, and the y-axis will show the CO2 emissions per person.
- The size of each point will represent the country's population.

```
co2_file = 'data/national/co2_emissions_tonnes_per_person.csv'
gdp_file = 'data/national/gdppercapita_us_inflation_adjusted.csv'
pop_file = 'data/national/population.csv'

co2_per_cap = pd.read_csv(co2_file, index_col=0, parse_dates=True)
gdp_per_cap = pd.read_csv(gdp_file, index_col=0, parse_dates=True)
population = pd.read_csv(pop_file, index_col=0, parse_dates=True)
```

Vector properties: example

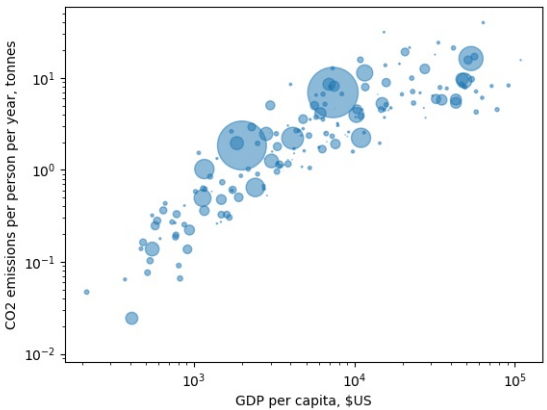
- `population`:

	Afghanistan	Albania	Algeria	Andorra	...
1960	9000000	1640000	11100000	13400	
1961	9170000	1690000	11300000	14400	
1962	9350000	1740000	11600000	15400	
1963	9540000	1790000	11900000	16400	
...					

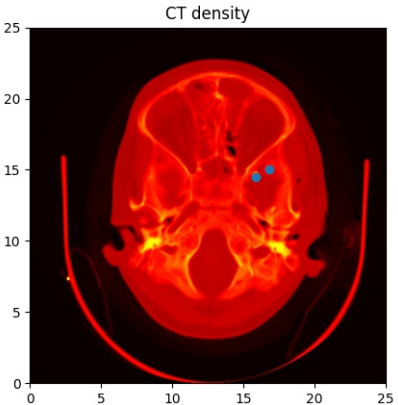
Vector properties: example

```
plt.figure()

x = gdp_per_cap.loc['2017']     # gdp in 2017
y = co2_per_person.loc['2017']  # co2 emmissions in 2017

# population in 2017 will give size of points (divide pop by 1M)
size = population.loc['2017'] / 1e6

# scatter points with vector size and some transparancy
plt.scatter(x, y, s=size, alpha=0.5)

# set a log-scale
plt.xscale('log')
plt.yscale('log')

plt.xlabel('GDP per capita, $US')
plt.ylabel('CO2 emissions per person per year, tonnes')
```

Vector properties: example

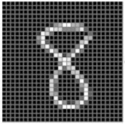


New topic: images and heat maps



Images and heat maps

- When a computer sees a grey-scale image, it really sees a 2x2 grid of numbers.



28 x 28
784 pixels

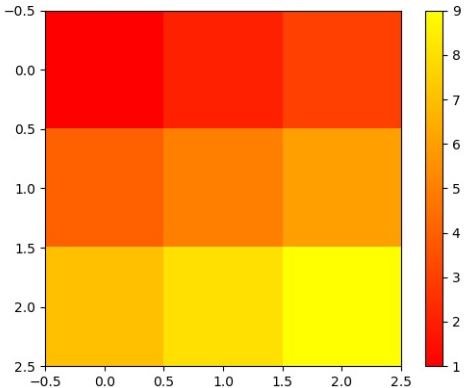
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Images and heat maps

- We could create a grid of numbers using lists. If we then pass this grid to the function `plt.imshow()`, it will display this "image". We can also choose the "`cmap`", which maps each number to a colour.

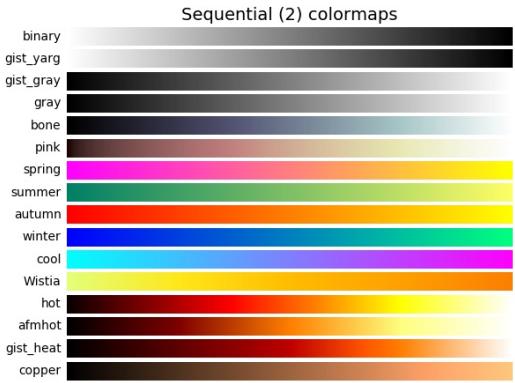
```
plt.figure()

# create grid of numbers
grid = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]

# plot the grid with 'autumn' color map
plt.imshow(grid, cmap='autumn')

# add a colour key
plt.colorbar()
```

Images and heat maps



Images and heat maps

- There are many colour maps to choose from, for example:



Images and heat maps

- One situation where `imshow()` can be useful is in visualising the pairwise correlation between a group of things.

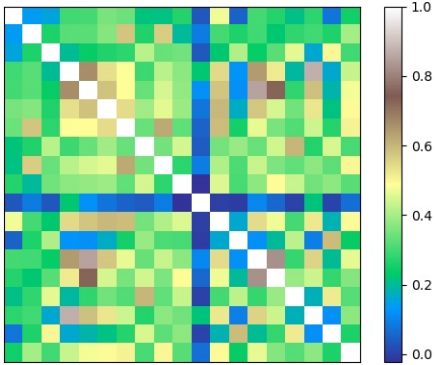
```
data = pd.read_csv("data/stocks/FTSE_stock_prices.csv", index_col=0)
correlation_matrix = data.pct_change().corr()
```

	AAL	AZN	BATS.L	BHP	...
AAL	1	0.13987	0.156111	0.298935	
AZN	0.13987	1	0.264094	0.3242	
BATS.L	0.156111	0.264094	1	0.201739	
BHP	0.298935	0.324262	0.201739	1	
...					

Images and heat maps

```
# create a new figure
plt.figure()

# imshow the grid of correlation
plt.imshow(correlation_matrix, cmap='terrain')

# add a color bar
plt.colorbar()

# remove cluttering x and y ticks
plt.xticks([])
plt.yticks([])
```

Images and heat maps



Images and heat maps: example

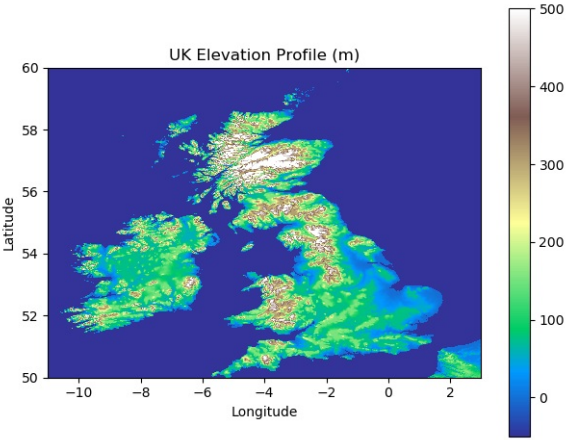
```
elevation = pd.read_csv('UK_elevation.csv', index_col=0)
```

	-4.1667	-4.1583	-4.15	-4.1417	-4.1333	...
57.4917	601	577	583	576	576	
57.4833	576	561	500	395	318	
57.475	491	516	339	322	423	
57.4667	507	491	500	516	549	
57.4583	573	576	586	596	613	
...						

Images and heat maps: example

```
# create figure
plt.figure()

# imshow data
plt.imshow(elevation,                # grid data
           vmin=-50,                 # minimum for colour bar
           vmax=500,                 # maximum for colour bar
           cmap='terrain',           # terrain style colour map
           extent=[-11, 3, 50, 60])  # [x1, x2, y1, y2] plot boundaries

# add axis labels and a title
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('UK Elevation Profile')

# add a colourbar
plt.colorbar()
```

Images and heat maps: example



Thanks!