Data Visualization With Python

June 14, 2018

GSA

D2D

DATA TO DECISIONS

Course Outline

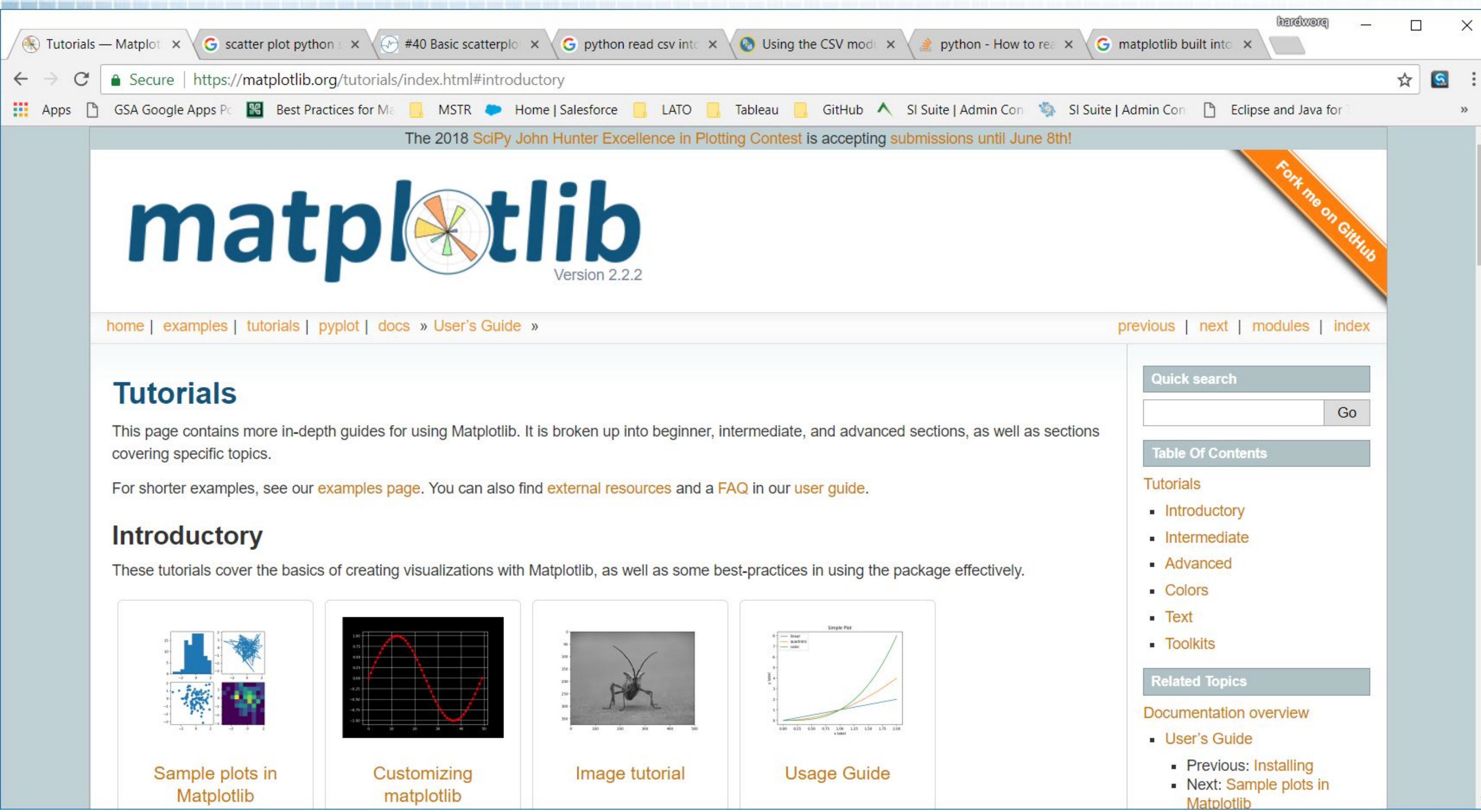
https://www.datacamp.com/courses/introduction-to-data-visualization-with-python

- **Python tools**
  - **Matplotlib**
  - **Seaborn**
- **Exercises**
  - **Plotting skills: common axes, axes in figure, subplots**
  - **Customization of plots**
  - **Regressions / residuals / grouping**
  - **Distributions: strip, swarm and violin plots**
- **Bonus**
  - **2D plots**
  - **Turtle**

GSA

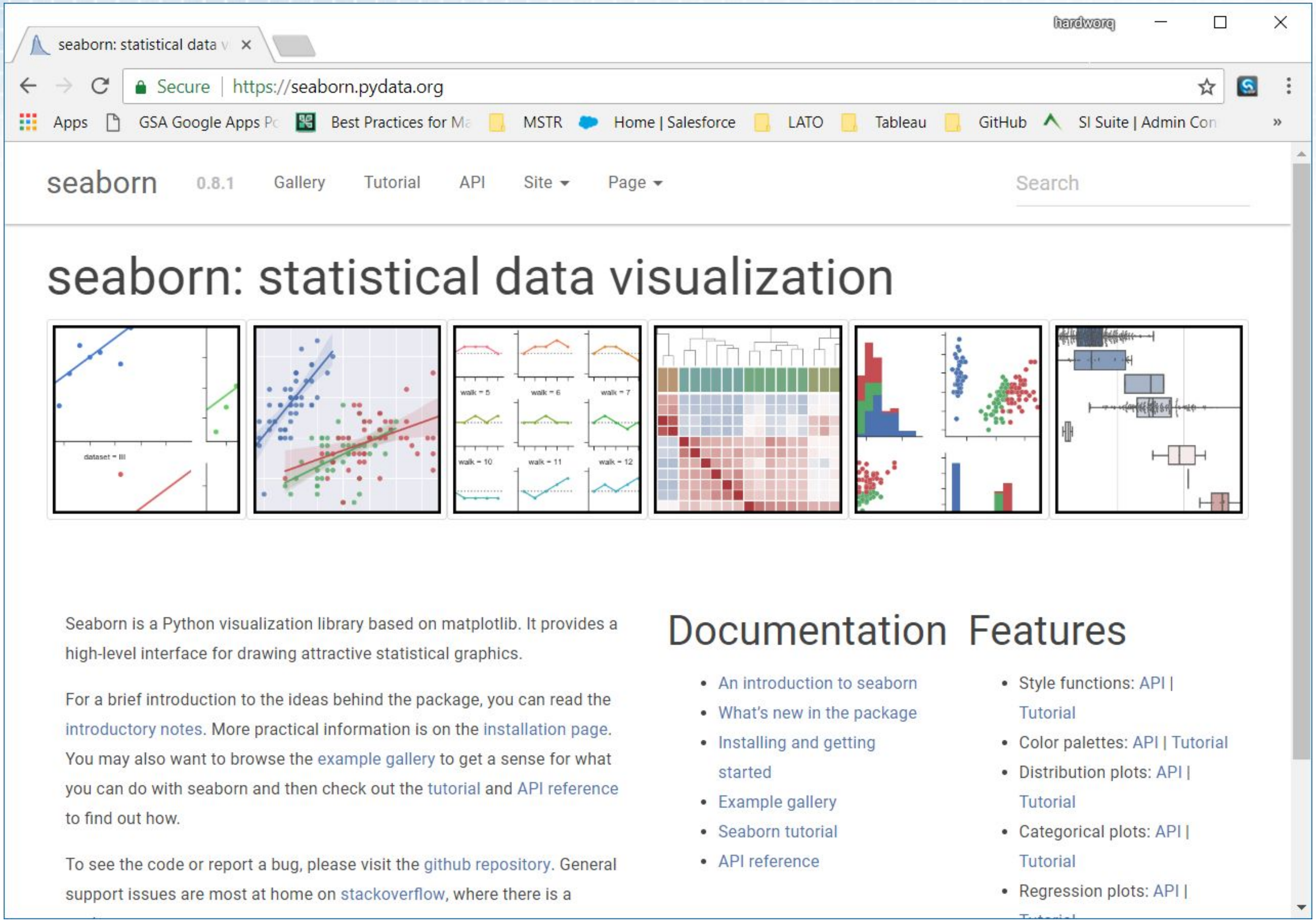
D2D
DATA TO DECISIONS

https://matplotlib.org/tutorials/index.html#introductory

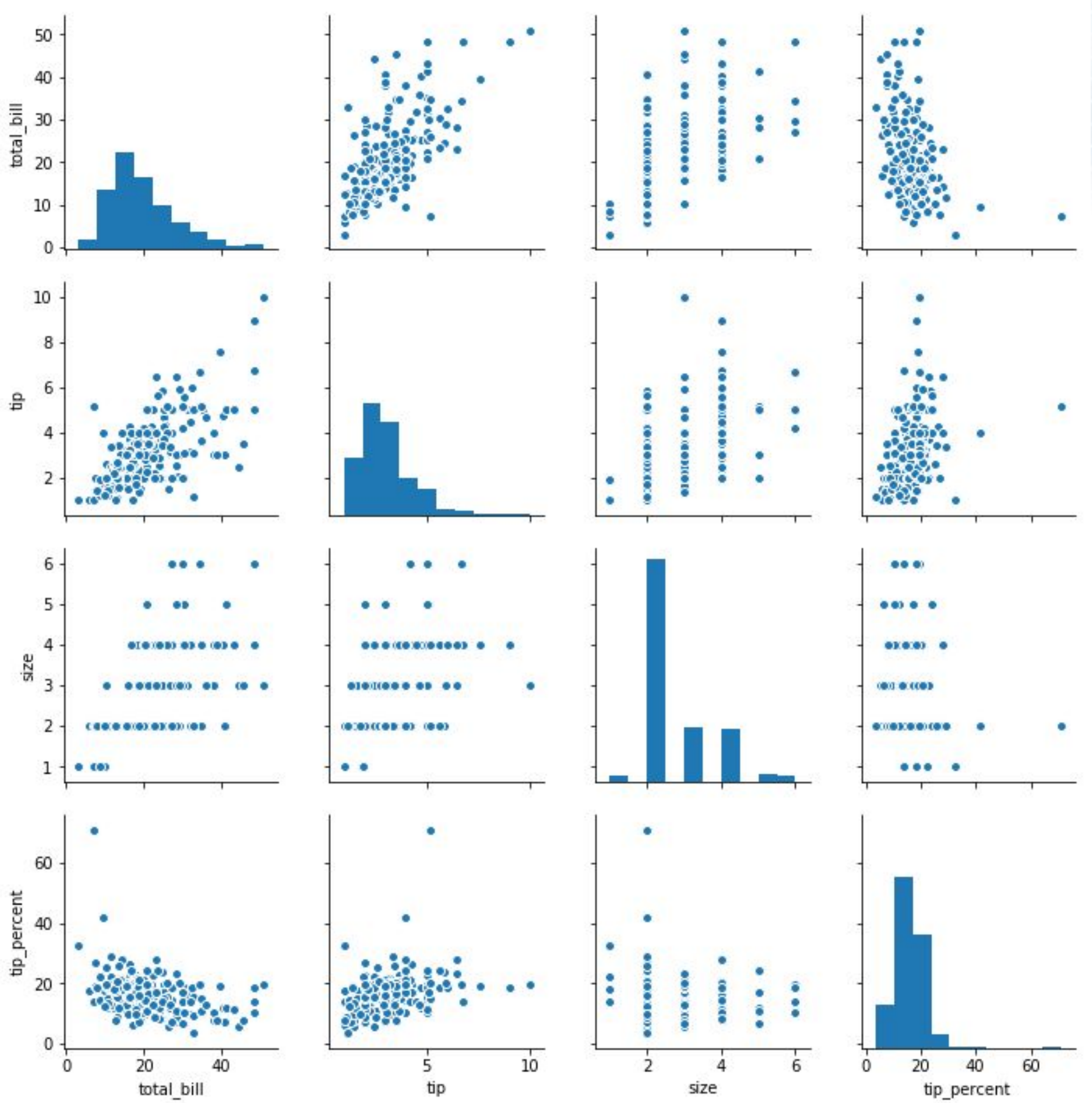


https://matplotlib.org/users/customizing.html
https://tonysyu.github.io/raw_content/matplotlib-style-gallery/gallery.html

https://seaborn.pydata.org/



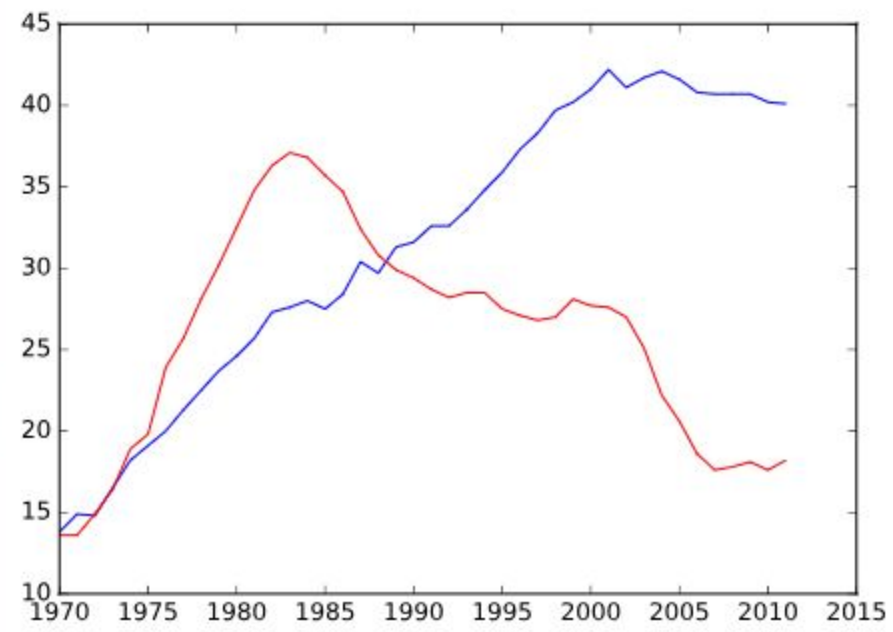
Copy / save plot image w/ right click



Customizing plots

Multiple plots on single axis

```
# Import matplotlib.pyplot
import matplotlib.pyplot as plt

# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')

# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year,computer_science, color='red')

# Display the plot
plt.show()

```

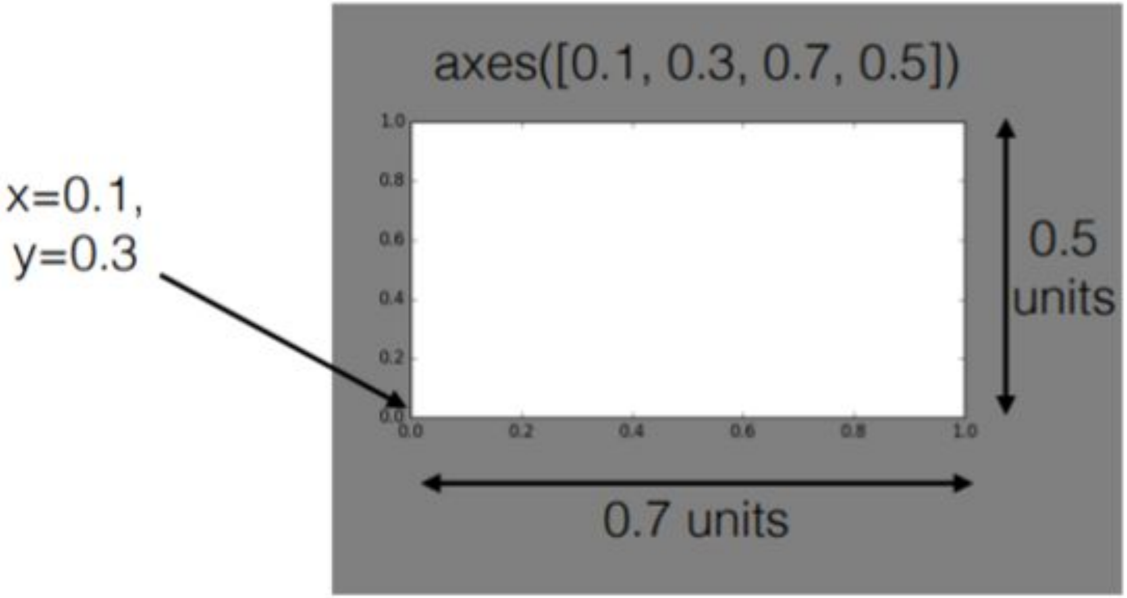


GSA

D2D
DATA TO DECISIONS

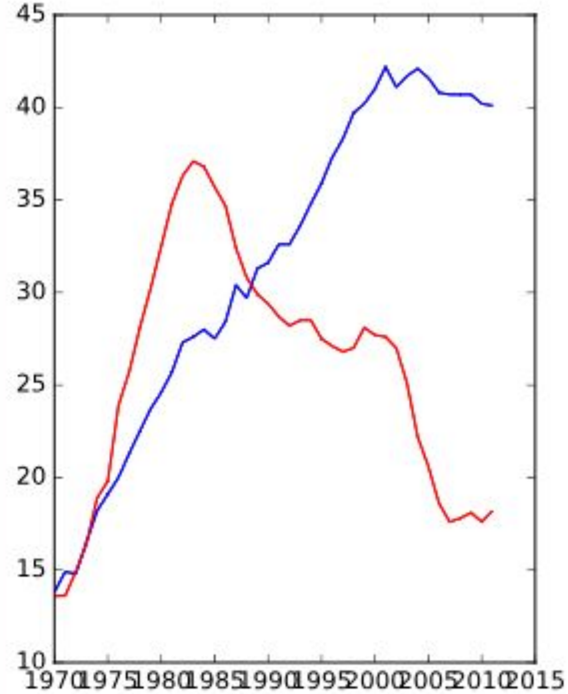
axes() command

- Syntax: axes([x_lo, y_lo, width, height])
- Units between 0 and 1 (figure dimensions)



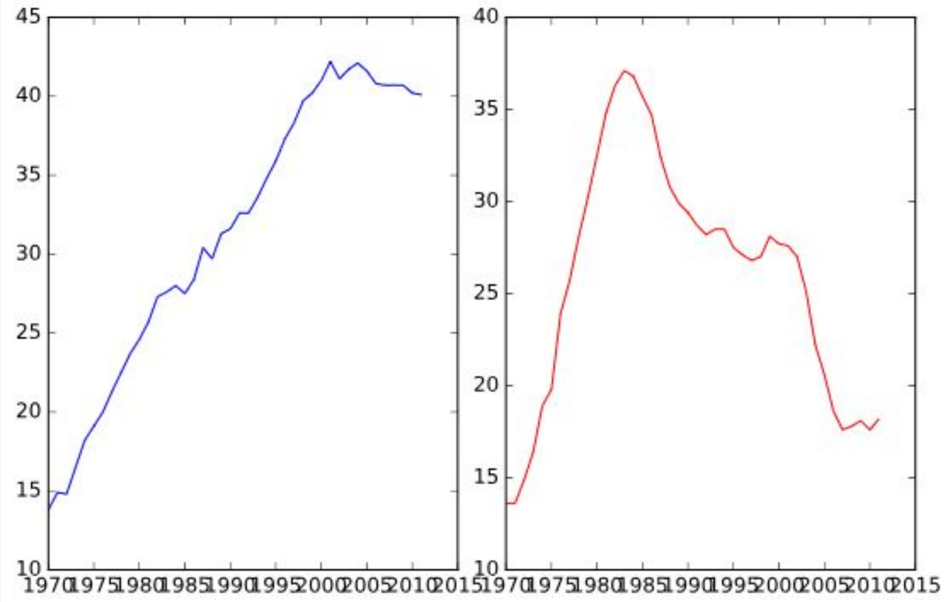
Using axes()

```python
# Create plot axes for the first line plot
plt.axes([0.05, 0.05, 0.425, 0.9])

# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')

# Create plot axes for the second line plot
plt.axes([0.05, 0.05, 0.425, 0.9])

# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year, computer_science, color='red')

# Display the plot
plt.show()
```



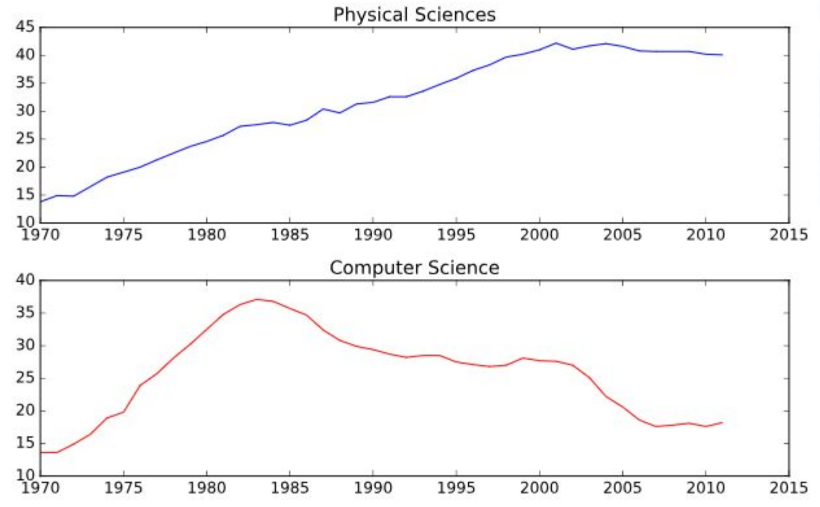
Using axes() (2)

```python
# Create plot axes for the first line plot
plt.axes([0.05, 0.05, 0.425, 0.9])

# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')

# Create plot axes for the second line plot
plt.axes([0.525, 0.05, 0.425, 0.9])

# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year, computer_science, color='red')

# Display the plot
plt.show()

```

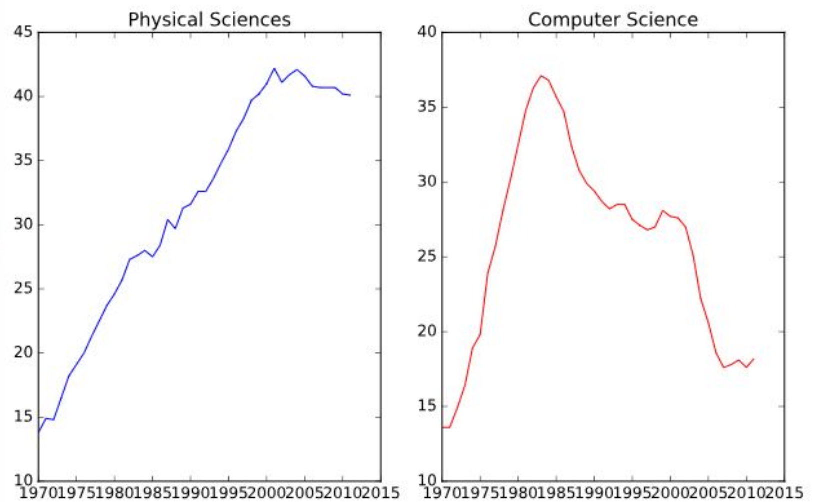


Using subplot() (1)

```
# Create a figure with 1x2 subplot and make the left subplot active
plt.subplot(2,1,1)

# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')
plt.title('Physical Sciences')

# Make the right subplot active in the current 1x2 subplot grid
plt.subplot(2,1,2)

# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year, computer_science, color='red')
plt.title('Computer Science')

```



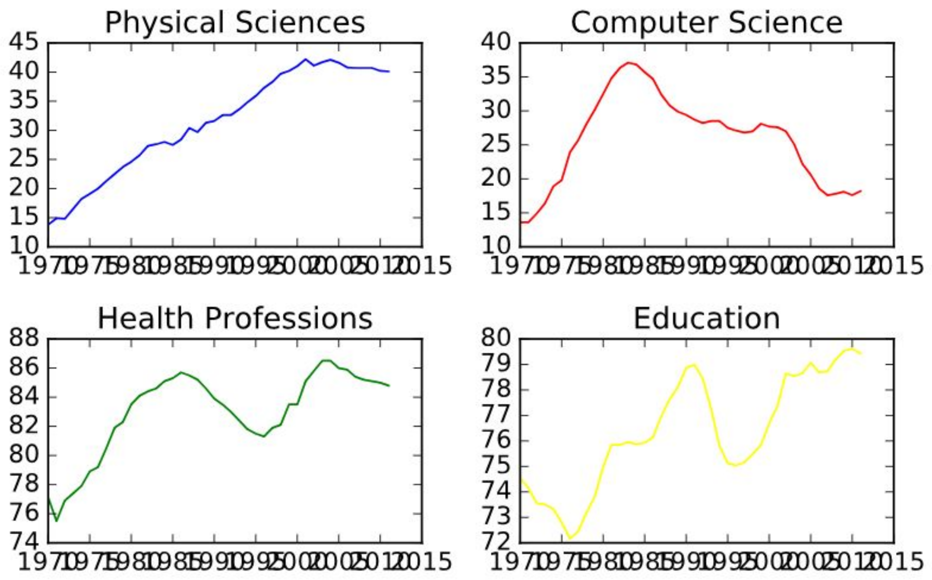

```
# Create a figure with 1x2 subplot and make the left subplot active
plt.subplot(1,2,1)

# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')
plt.title('Physical Sciences')

# Make the right subplot active in the current 1x2 subplot grid
plt.subplot(1,2,2)

# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year, computer_science, color='red')
plt.title('Computer Science')

# Use plt.tight_layout() to improve the spacing between subplots
plt.tight_layout()
plt.show()

```



Using subplot() (2)

- Syntax: subplot(nrows, ncols, nsubplot)
- Subplot ordering:
  - Row-wise from top left
  - Indexed from 1

```
# Create a figure with 2x2 subplot layout and make the top left subplot active
plt.subplot(2, 2, 1)
# Plot in blue the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')
plt.title('Physical Sciences')
# Make the top right subplot active in the current 2x2 subplot grid
plt.subplot(2, 2, 2)
# Plot in red the % of degrees awarded to women in Computer Science
plt.plot(year, computer_science, color='red')
plt.title('Computer Science')
# Make the bottom left subplot active in the current 2x2 subplot grid
plt.subplot(2, 2, 3)
# Plot in green the % of degrees awarded to women in Health Professions
plt.plot(year, health, color='green')
plt.title('Health Professions')
# Make the bottom right subplot active in the current 2x2 subplot grid
plt.subplot(2, 2, 4)
# Plot in yellow the % of degrees awarded to women in Education
plt.plot(year, education, color='yellow')
plt.title('Education')
# Improve the spacing between subplots and display them
plt.tight_layout()
plt.show()
```



Customizing axes

Controlling axis extents

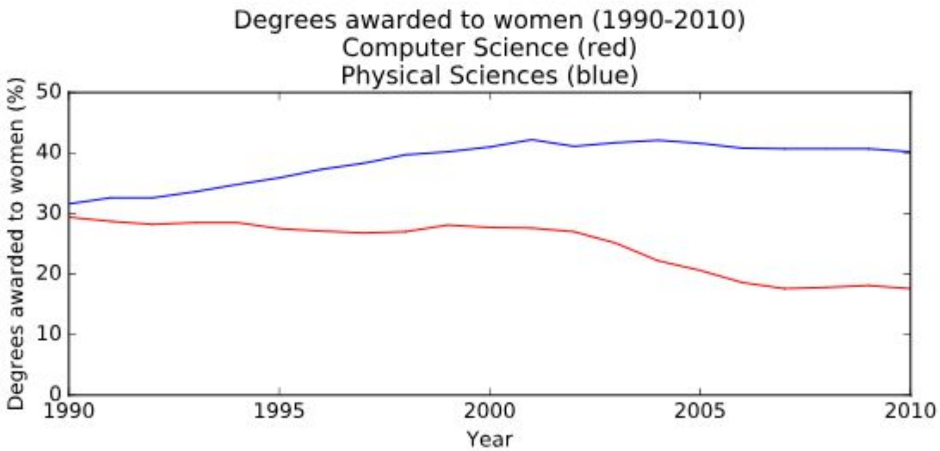
- axis([xmin, xmax, ymin, ymax]) sets axis extents
- Control over individual axis extents
  - xlim([xmin, xmax])
  - ylim([ymin, ymax])
- Can use tuples, lists for extents
  - e.g., xlim((-2, 3)) works
  - e.g., xlim([-2, 3]) works also

Other axis() options

Invocation	Result
axis('off')	turns off axis lines, labels
axis('equal')	equal scaling on x, y axes
axis('square')	forces square plot
axis('tight')	sets xlim(), ylim() to show all data

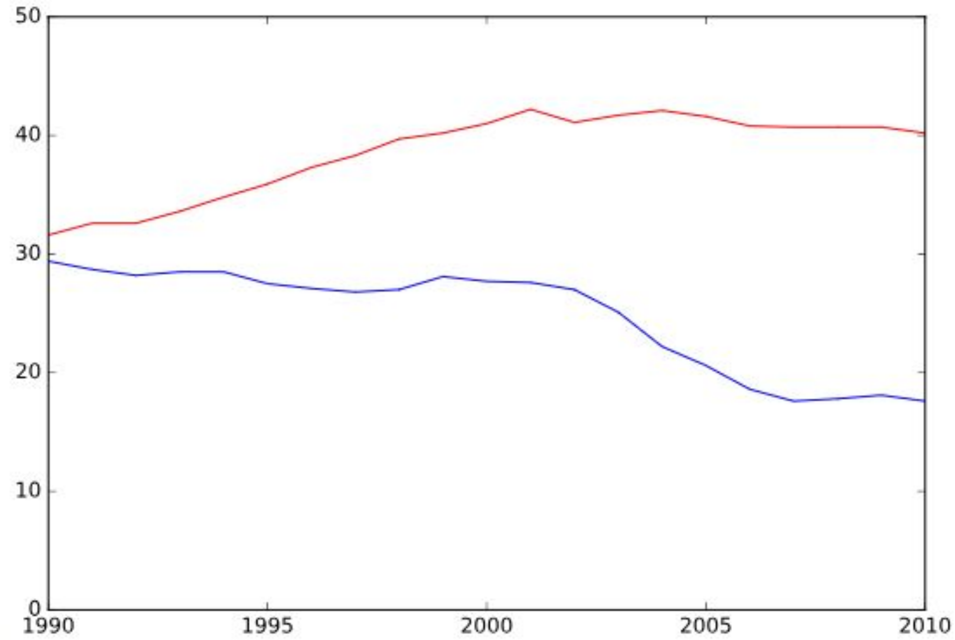
Using xlim(), ylim()

```
# Plot the % of degrees awarded to women in Computer Science and the Physical Sciences
plt.plot(year,computer_science, color='red')
plt.plot(year, physical_sciences, color='blue')

# Add the axis labels
plt.xlabel('Year')
plt.ylabel('Degrees awarded to women (%)')

# Set the x-axis range
plt.xlim(1990, 2010)

# Set the y-axis range
plt.ylim(0, 50)

# Add a title and display the plot
plt.title('Degrees awarded to women (1990-2010)\nComputer Science (red)\nPhysical Sciences (blue)')
plt.show()

# Save the image as 'xlim_and_ylim.png'
plt.savefig('xlim_and_ylim.png')
```



Use plt.axis()

```
# Plot in blue the % of degrees awarded to women in Computer Science
plt.plot(year,computer_science, color='blue')

# Plot in red the % of degrees awarded to women in the Physical Sciences
plt.plot(year, physical_sciences,color='red')

# Set the x-axis and y-axis limits
plt.axis((1990,2010,0,50))

# Show the figure
plt.show()

# Save the figure as 'axis_limits.png'
plt.savefig('axis_limits.png')

```



Legend

Legend locations

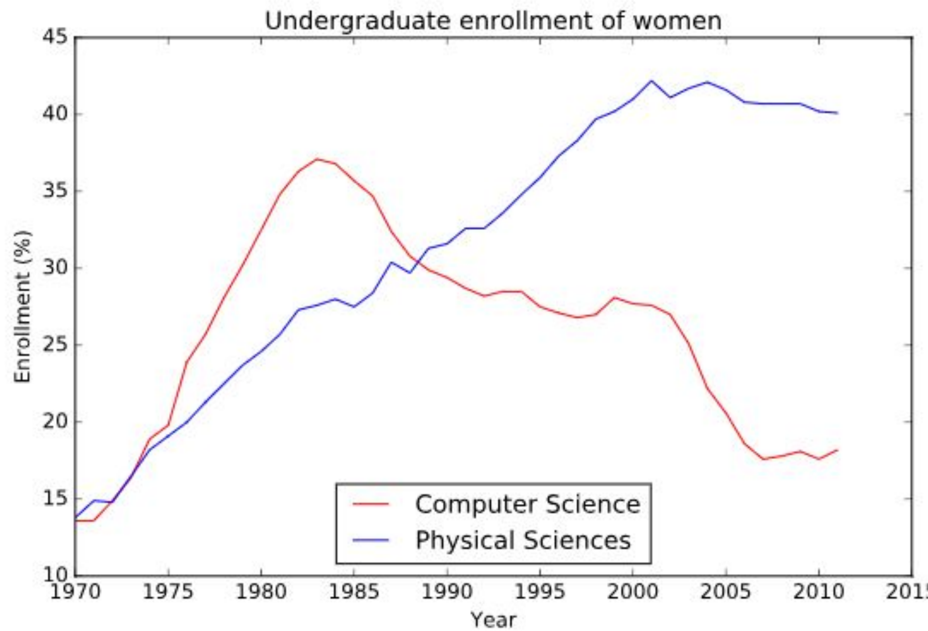
string	code	string	code	string	code
'upper left'	2	'upper center'	9	'upper right'	1
'center left'	6	'center'	10	'center right'	7
'lower left'	3	'lower center'	8	'lower right'	4
'best'	0			'right'	5

Options for annotate()

option	description
s	text of label
xy	coordinates to annotate
xytext	coordinates of label
arrowprops	controls drawing of arrow

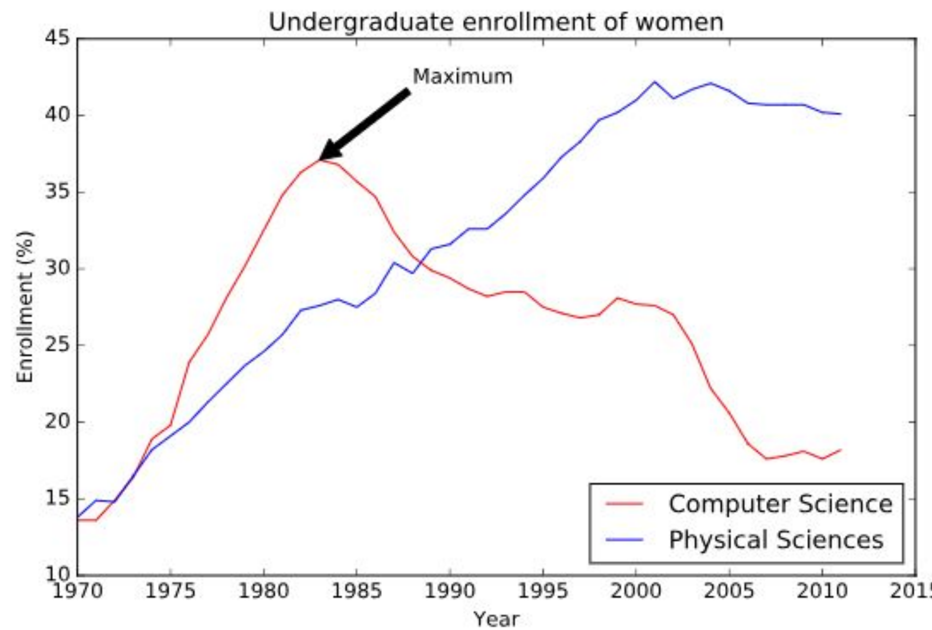
Legend (Cont. 2)

```python
# Specify the label 'Computer Science'
plt.plot(year, computer_science, color='red', label
='Computer Science')

# Specify the label 'Physical Sciences'
plt.plot(year, physical_sciences, color='blue', label
='Physical Sciences')

# Add a legend at the lower center
plt.legend()

# Add axis labels and title
plt.xlabel('Year')
plt.ylabel('Enrollment (%)')
plt.title('Undergraduate enrollment of women')
plt.legend(loc = 'lower center')
plt.show()
```



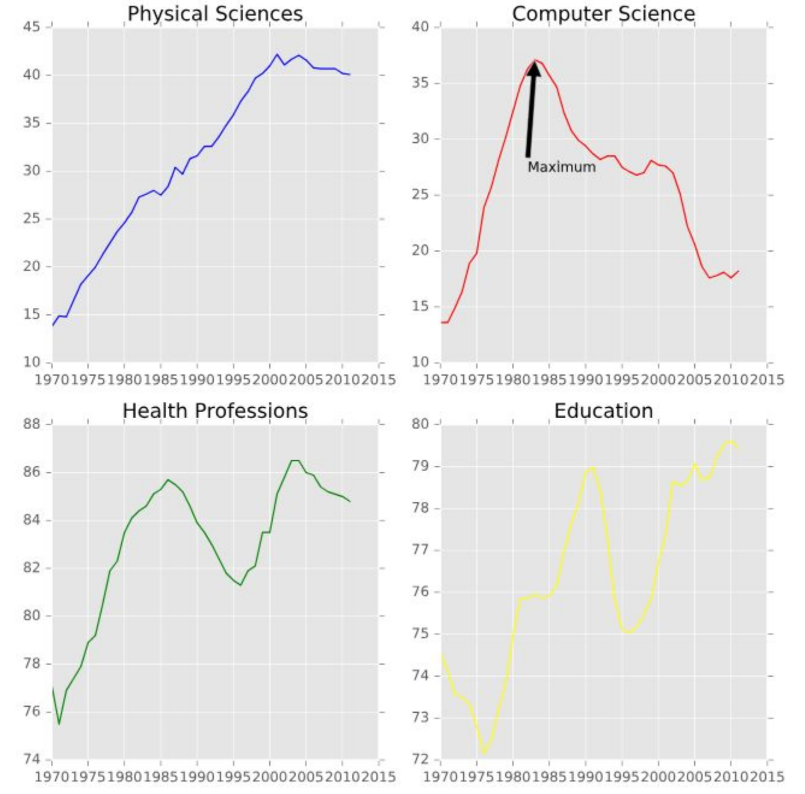
Using annotate()

```
# Plot with legend as before
plt.plot(year, computer_science, color='red', label='Computer Science')
plt.plot(year, physical_sciences, color='blue', label='Physical Sciences')
plt.legend(loc='lower right')

# Compute the maximum enrollment of women in Computer Science: cs_max
cs_max = computer_science.max()

# Calculate the year in which there was maximum enrollment of women in Computer Science: yr_max
yr_max = year[computer_science.argmax()]

# Add a black arrow annotation
plt.annotate('Maximum', xy=(yr_max, cs_max), xytext=(yr_max+5, cs_max+5), arrowprops=dict(facecolor='black'))

# Add axis labels and title
plt.xlabel('Year')
plt.ylabel('Enrollment (%)')
plt.title('Undergraduate enrollment of women')
plt.show()
```



Modifying styles

```python
# Import matplotlib.pyplot
import matplotlib.pyplot as plt
# Set the style to 'ggplot'
plt.style.use('ggplot')
# Create a figure with 2x2 subplot layout
plt.subplot(2, 2, 1)
# Plot the enrollment % of women in the Physical Sciences
plt.plot(year, physical_sciences, color='blue')
plt.title('Physical Sciences')
# Plot the enrollment % of women in Computer Science
plt.subplot(2, 2, 2)
plt.plot(year, computer_science, color='red')
plt.title('Computer Science')
# Add annotation
cs_max = computer_science.max()
yr_max = year[computer_science.argmax()]
plt.annotate('Maximum', xy=(yr_max, cs_max), xytext=(yr_max-1, cs_max-10), arrowprops=dict
(facecolor='black'))
# Plot the enrollmment % of women in Health professions
plt.subplot(2, 2, 3)
plt.plot(year, health, color='green')
plt.title('Health Professions')
# Plot the enrollment % of women in Education
plt.subplot(2, 2, 4)
plt.plot(year, education, color='yellow')
plt.title('Education')
# Improve spacing between subplots and display them
plt.tight_layout()
plt.show()
```



- Style sheets in Matplotlib
- Defaults for lines, points, backgrounds, etc.
- Switch styles globally with plt.style.use()
- plt.style.available: list of styles

Statistical Plotting

GSA

D2D

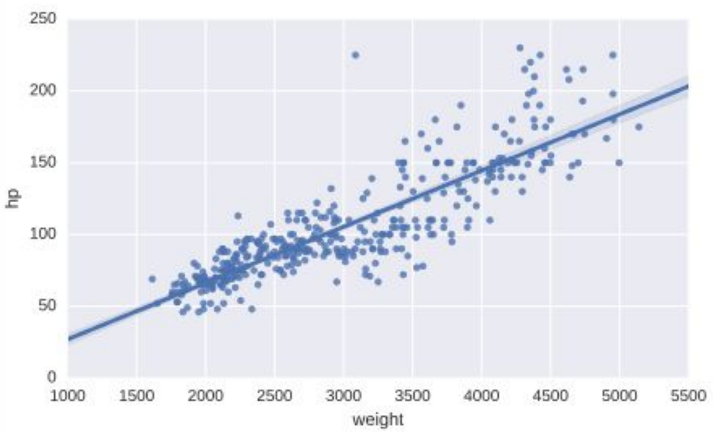
DATA TO DECISIONS

Seaborn

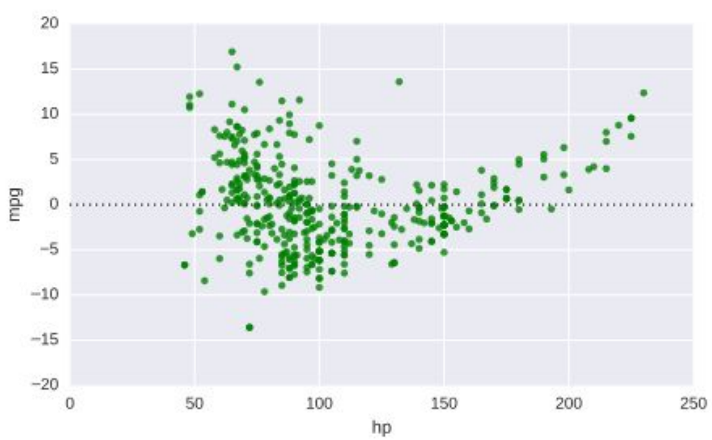
- Built on top of Matplotlib
- Purpose: simplify visualizing statistical data
- Works with Pandas – data frame package

Simple linear regressions and residuals

```python
# Import plotting modules
import matplotlib.pyplot as plt
import seaborn as sns

# Plot a linear regression between 'weight' and 'hp'
sns.lmplot(x ='weight', y ='hp', data = auto)

# Display the plot
plt.show()
```

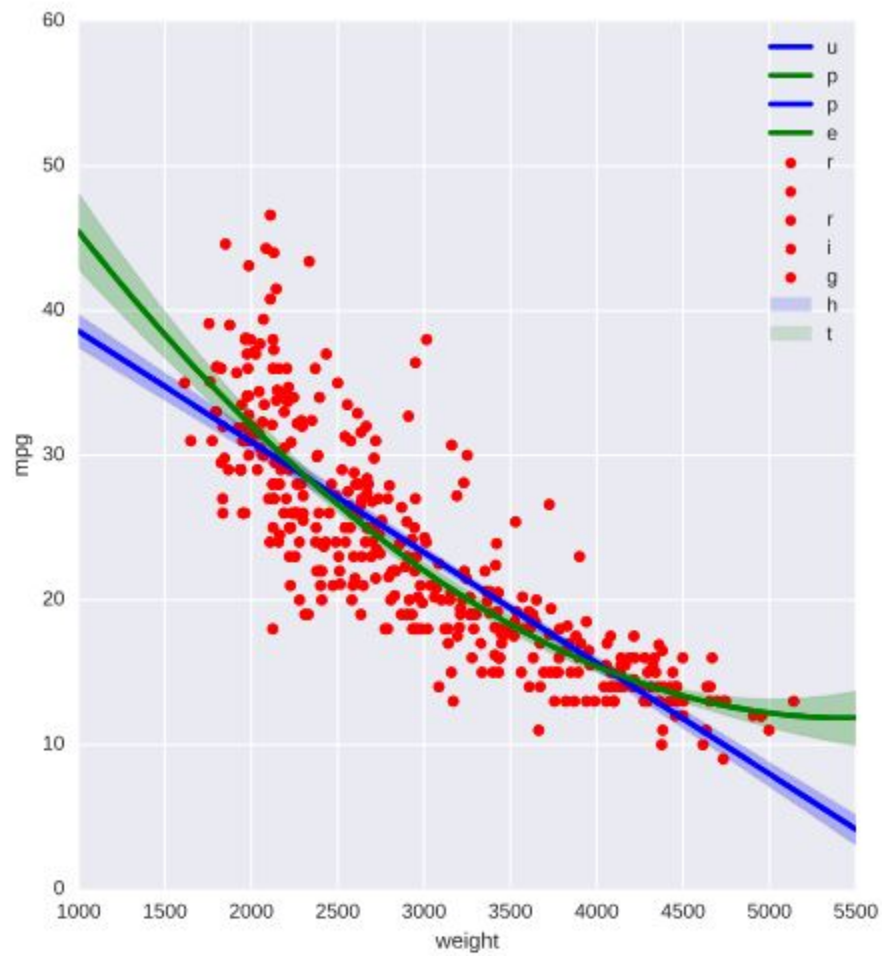



```python
# Import plotting modules
import matplotlib.pyplot as plt
import seaborn as sns

# Generate a green residual plot of the regression between 'hp' and 'mpg'
sns.residplot(x ='hp', y ='mpg', data = auto, color ='green')

# Display the plot
plt.show()
```



Higher-order regressions

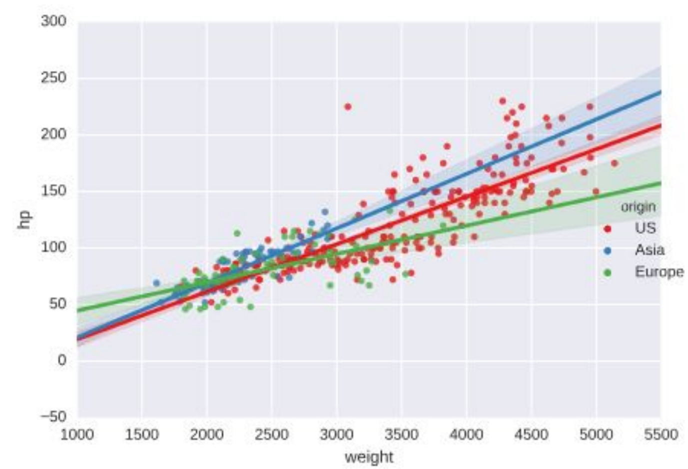
```
# Generate a scatter plot of 'weight' and 'mpg' using red circles
plt.scatter(auto['weight'], auto['mpg'], label='data', color='red', marker='o')

# Plot in blue a linear regression of order 1 between 'weight' and 'mpg'
sns.regplot(x='weight', y='mpg', data=auto, scatter=None, color='blue', label='order 1')

# Plot in green a linear regression of order 2 between 'weight' and 'mpg'
sns.regplot(x='weight', y='mpg', data=auto, scatter=None, order=2, color='green', label='order 2')

# Add a legend and display the plot
plt.legend(loc='upper right')
plt.show()
```



Grouping linear regressions

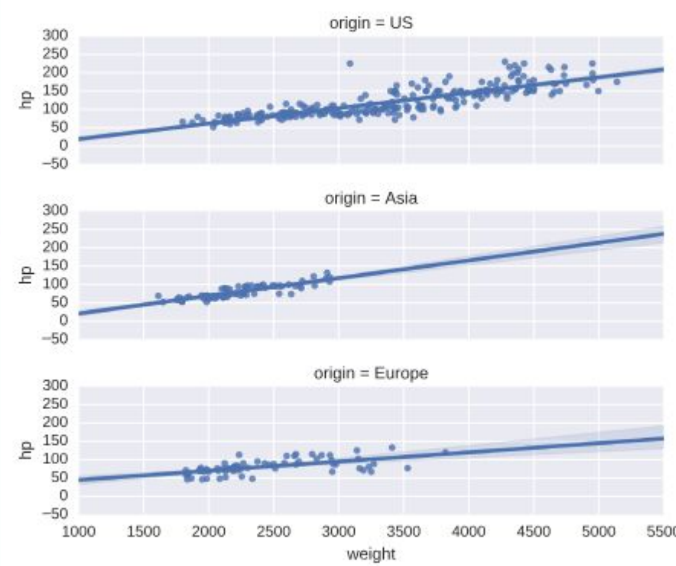
by hue

```
# Plot a linear regression between 'weight' and 'hp', with a hue of 'origin' and palette of 'Set1'
sns.lmplot(x='weight', y='hp', data=auto, hue='origin', palette='Set1')

# Display the plot
plt.show()
```



by row / column

```
# Plot linear regressions between 'weight' and 'hp' grouped row-wise by 'origin'
sns.lmplot(x ='weight', y ='hp', data=auto, row='origin', palette='Set1')

# Display the plot
plt.show()
```

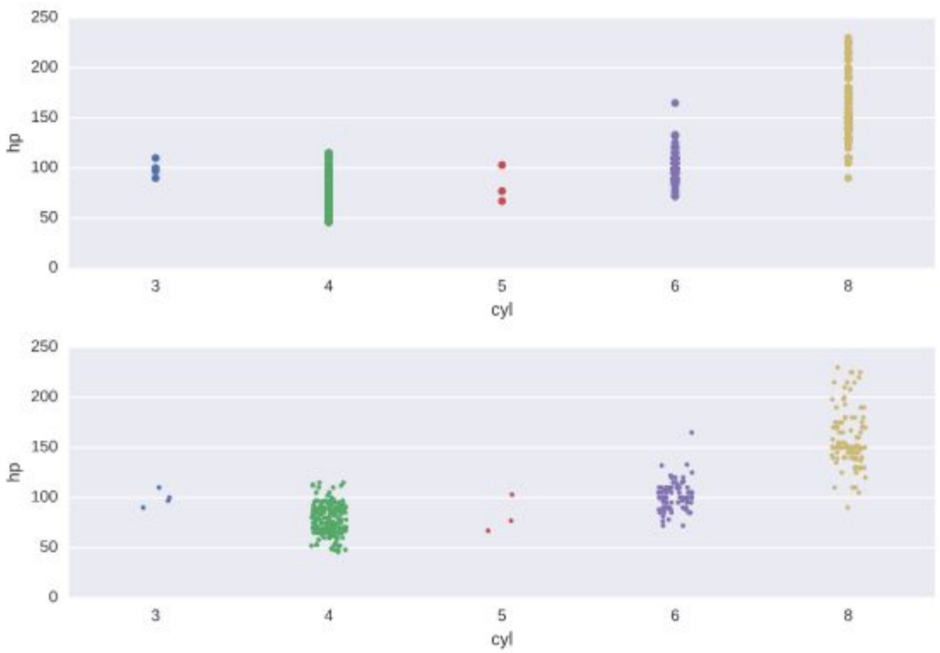


GSA

D2D
DATA TO DECISIONS

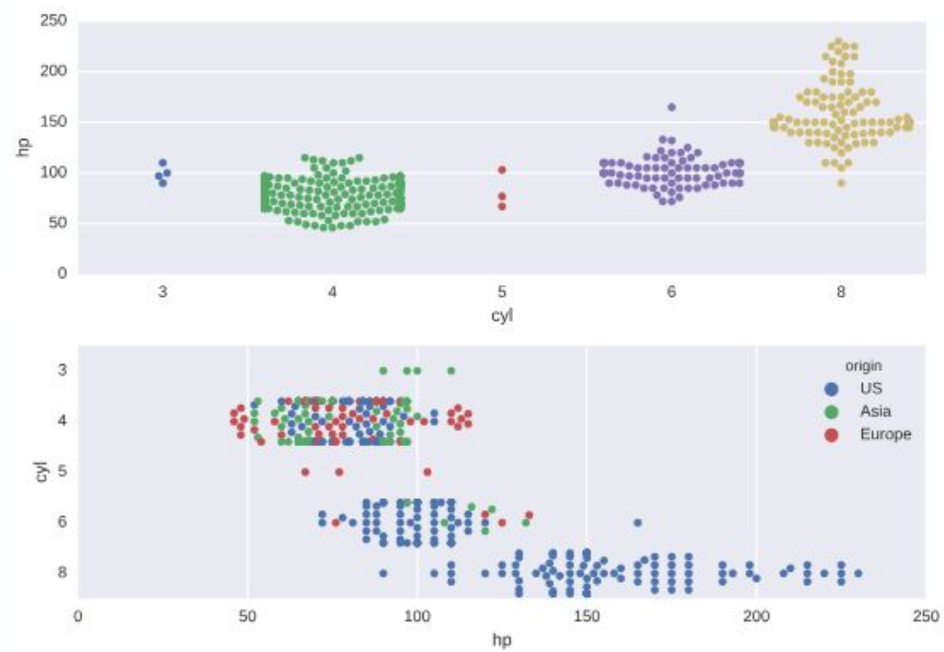
Constructing strip plots

```
# Make a strip plot of 'hp' grouped by 'cyl'
plt.subplot(2,1,1)
sns.stripplot(x='cyl', y='hp', data=auto)

# Make the strip plot again using jitter and a smaller point size
plt.subplot(2,1,2)
sns.stripplot(x='cyl', y='hp', data=auto, jitter = True, size = 3)

# Display the plot
plt.show()

```



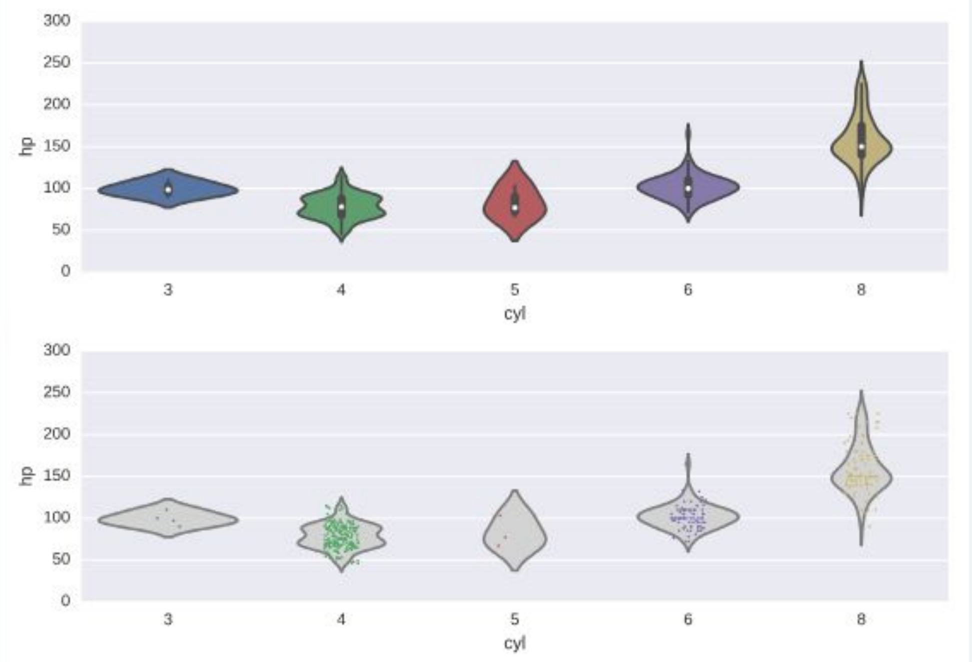
Constructing swarm plots

```
# Generate a swarm plot of 'hp' grouped horizontally by
'cyl'
plt.subplot(2,1,1)
sns.swarmplot(x = 'cyl', y = 'hp', data = auto )

# Generate a swarm plot of 'hp' grouped vertically by 'cyl'
with a hue of 'origin'
plt.subplot(2,1,2)
sns.swarmplot(x = 'hp', y = 'cyl', data = auto, orient =
'h', hue = 'origin')

# Display the plot
plt.show()
```



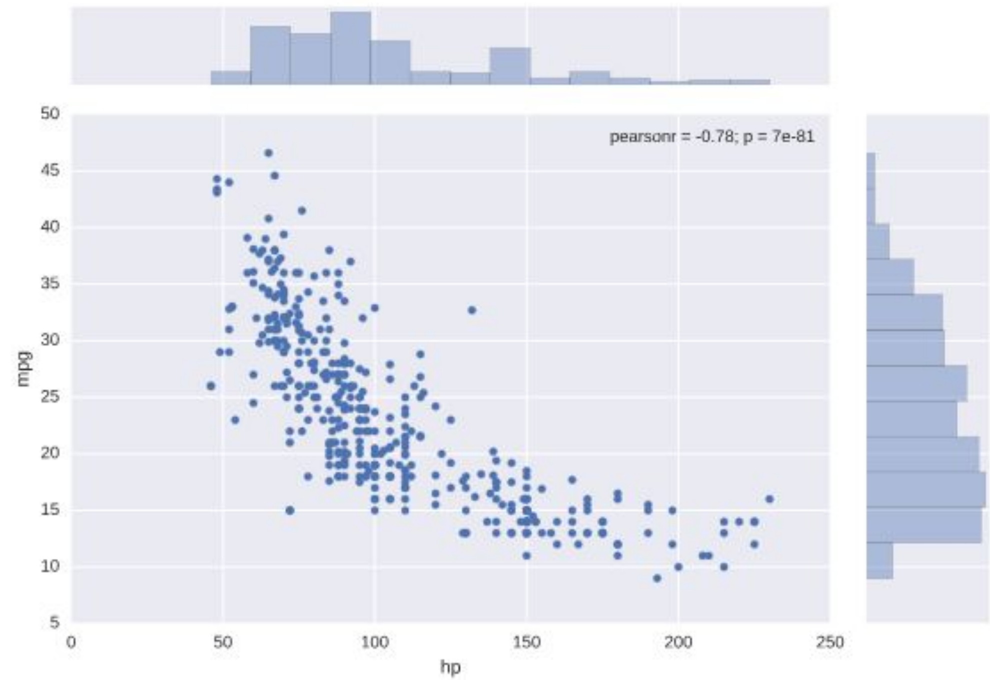
Constructing violin plots

```
# Generate a violin plot of 'hp' grouped horizontally by 'cyl'
plt.subplot(2,1,1)
sns.violinplot(x='cyl', y='hp', data = auto)

# Generate the same violin plot again with a color of 'lightgray' and without inner annotations
plt.subplot(2,1,2)
sns.violinplot(x='cyl', y='hp', data = auto, inner = None, color = 'lightgray')

# Overlay a strip plot on the violin plot
plt.subplot(2,1,2)
sns.stripplot(x='cyl', y='hp', data=auto, jitter = True, size = 1.5)

# Display the plot
plt.show()
```



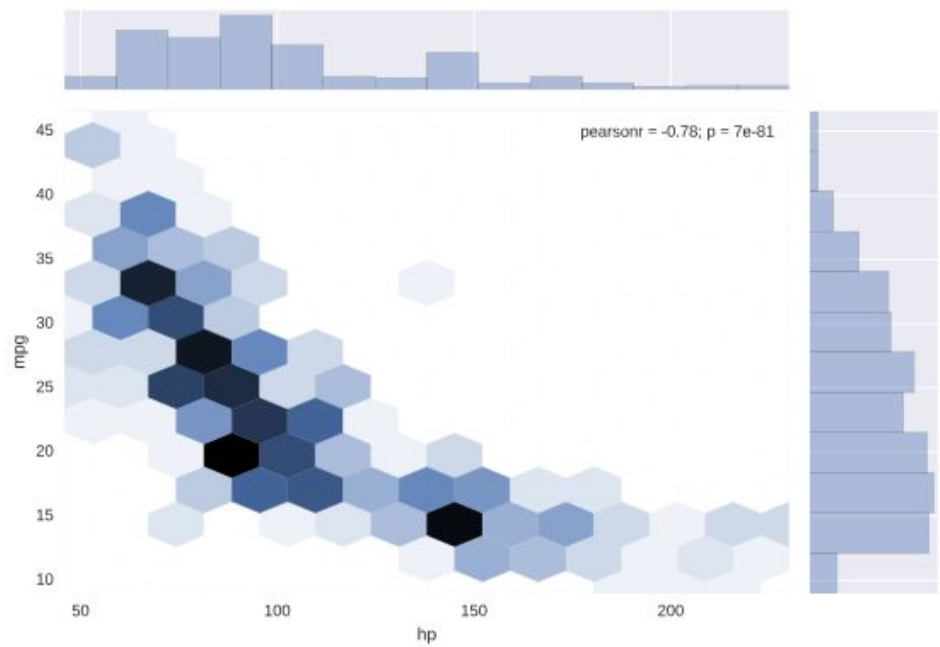
Plotting joint distributions (1)

```
# Generate a joint plot of 'hp' and 'mpg'
sns.jointplot(x = 'hp', y = 'mpg', data = auto)

# Display the plot
plt.show()
```



Plotting joint distributions (2)

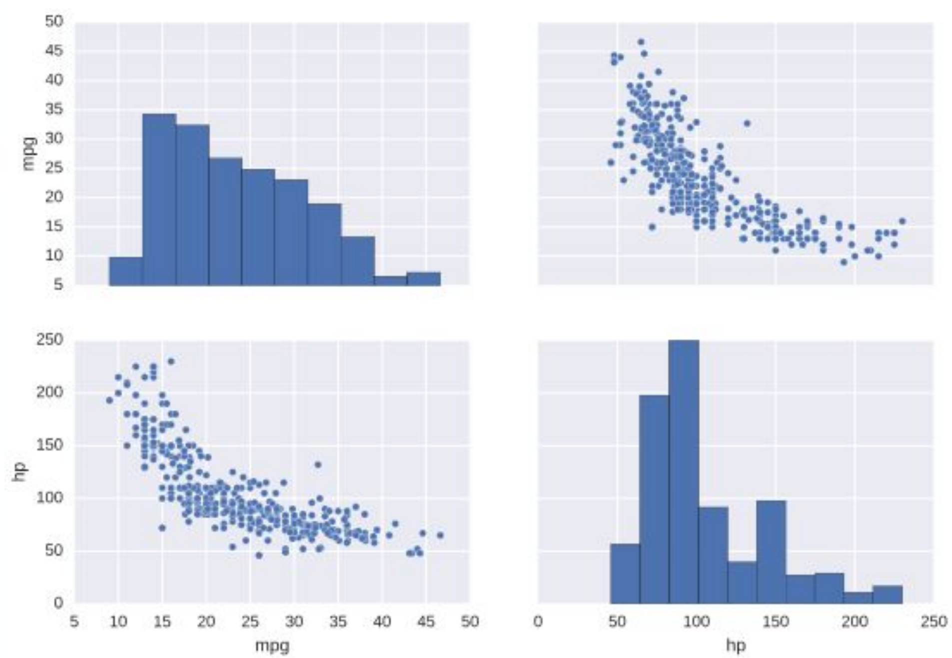
```
# Generate a joint plot of 'hp' and 'mpg' using a hexbin plot
sns.jointplot(x = 'hp', y = 'mpg', data = auto, kind='hex')

# Display the plot
plt.show()

```



- `kind='scatter'` uses a scatter plot of the data points
- `kind='reg'` uses a regression plot (default order 1)
- `kind='resid'` uses a residual plot
- `kind='kde'` uses a *kernel density estimate* of the joint distribution
- `kind='hex'` uses a hexbin plot of the joint distribution

Plotting distributions pairwise (1)

```python
# Print the first 5 rows of the DataFrame
print(auto.head())

# Plot the pairwise joint distributions from the DataFrame
sns.pairplot(auto)

# Display the plot
plt.show()

```

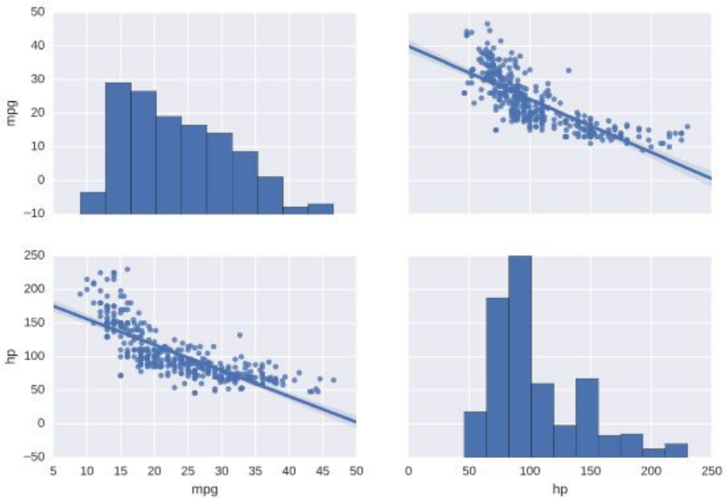


GSA

D2D
DATA TO DECISIONS

Plotting distributions pairwise (2)

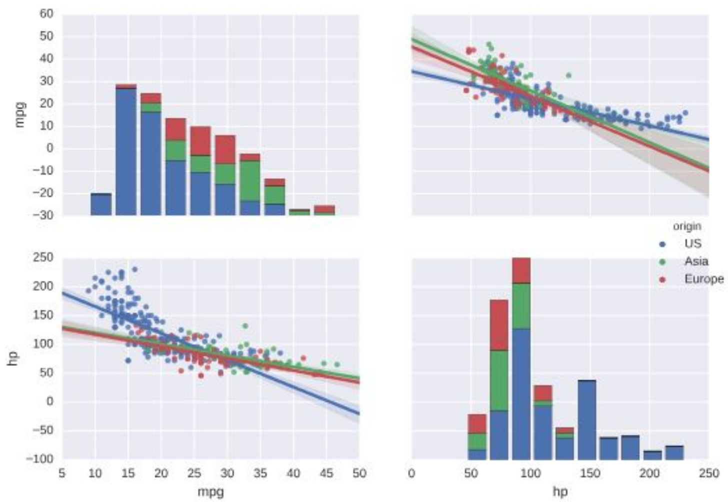
```python
# Print the first 5 rows of the DataFrame
print(auto.head())

# Plot the pairwise joint distributions grouped by 'origin'
along with regression lines
sns.pairplot(auto, kind='reg')

# Display the plot
plt.show()

```



<script.py> output:

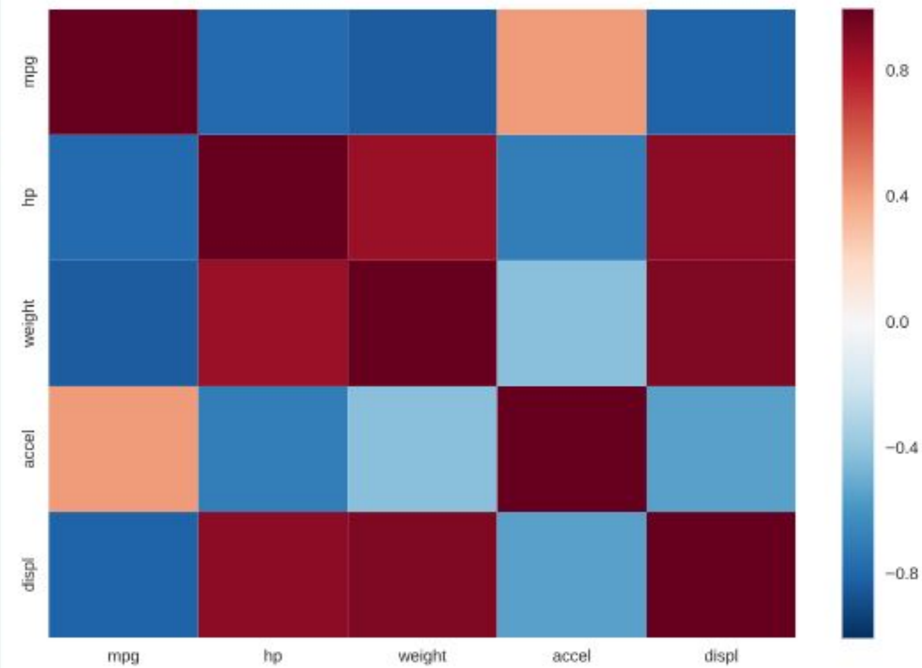
	mpg	hp	origin
0	18.0	88	US
1	9.0	193	US
2	36.1	60	Asia
3	18.5	98	US
4	34.3	78	Europe

```python
# Print the first 5 rows of the DataFrame
print(auto.head())

# Plot the pairwise joint distributions grouped by 'origin'
along with regression lines
sns.pairplot(auto, kind='reg', hue = 'origin')

# Display the plot
plt.show()

```



Visualizing correlations with a heatmap

```python
# Print the covariance matrix
print(cov_matrix)

# Visualize the covariance matrix using a heatmap
sns.heatmap(cov_matrix)

# Display the heatmap
plt.show()
```



	mpg	hp	weight	accel	displ
mpg	1.000000	-0.778427	-0.832244	0.423329	-0.805127
hp	-0.778427	1.000000	0.864538	-0.689196	0.897257
weight	-0.832244	0.864538	1.000000	-0.416839	0.932994
accel	0.423329	-0.689196	-0.416839	1.000000	-0.543800
displ	-0.805127	0.897257	0.932994	-0.543800	1.000000

GSA

D2D
DATA TO DECISIONS

Plotting 2D arrays

Reminder: NumPy arrays

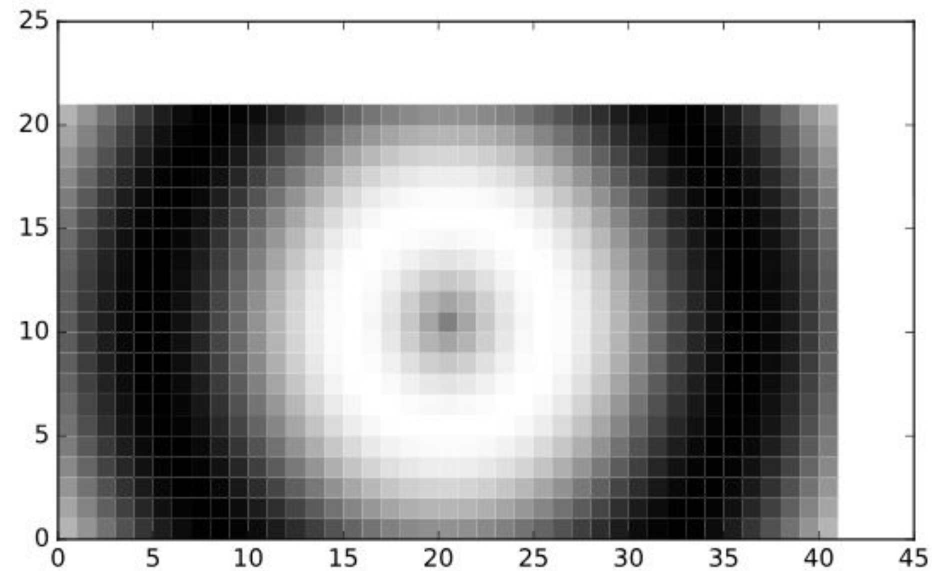
- Homogeneous in type
- Calculations all at once
- Indexing with brackets:
  - A[*index*] for 1D array
  - A[*index0*, *index1*] for 2D array

Reminder: slicing arrays

- Slicing: 1D arrays: A[*slice*], 2D arrays: A[*slice0*, *slice1*]
- Slicing: *slice* = *start*:*stop*:*stride*
  - Indexes from *start* to *stop*-*1* in steps of *stride*
  - Missing *start*: implicitly at *beginning* of array
  - Missing *stop*: implicitly at *end* of array
  - Missing *stride*: implicitly *stride 1*
- Negative indexes/slices: count from *end of array*

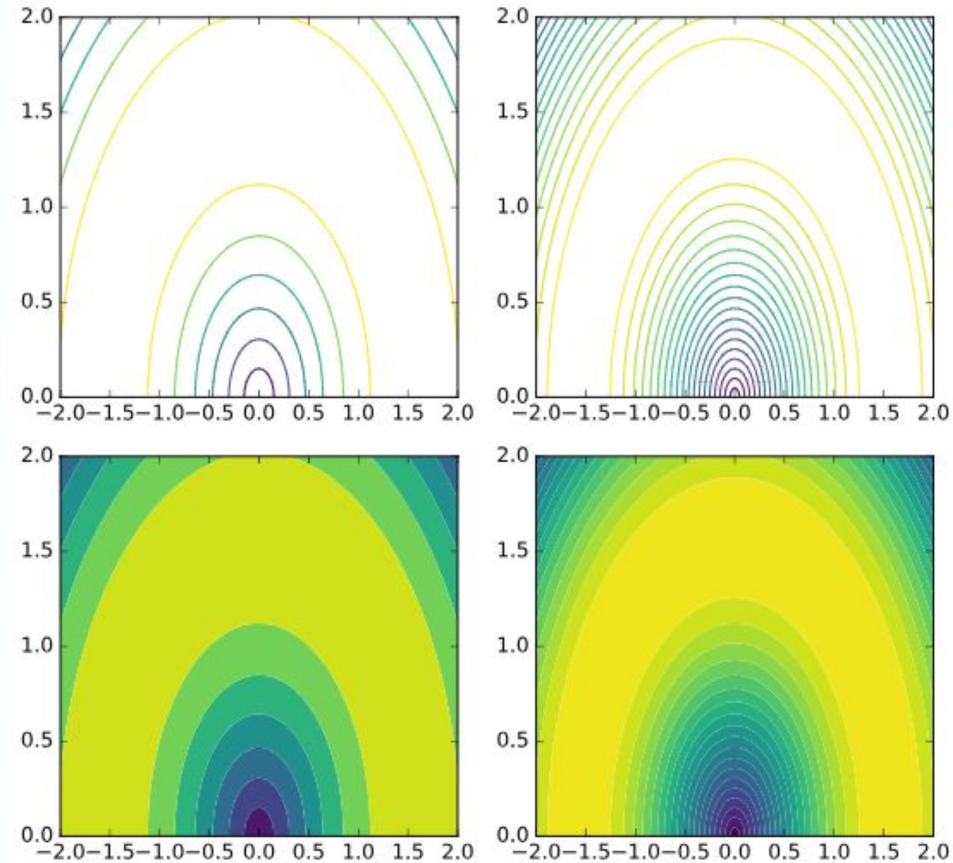
Generating meshes

```python
# Import numpy and matplotlib.pyplot
import numpy as np
import matplotlib.pyplot as plt
# Generate two 1-D arrays: u, v
u = np.linspace(-2, + 2, num = 41)
v = np.linspace(-1, + 1, num = 21)
# Generate 2-D arrays from u and v: X, Y
X,Y = np.meshgrid(u, v)
# Compute Z based on X and Y
Z = np.sin(3*np.sqrt(X**2 + Y**2))
# Display the resulting image with pcolor()
print ('Z:\n', Z)
plt.set_cmap('gray')
plt.pcolor(Z)
plt.show()
# Save the figure to 'sine_mesh.png'
plt.savefig('sine_mesh.png')

```



Contour & filled contour plots

```
# Generate a default contour map of the array Z
plt.subplot(2,2,1)
plt.contour(X, Y, Z)

# Generate a contour map with 20 contours
plt.subplot(2,2,2)
plt.contour(X, Y, Z, 20)

# Generate a default filled contour map of the array Z
plt.subplot(2,2,3)
plt.contourf(X, Y, Z)

# Generate a default filled contour map with 20 contours
plt.subplot(2,2,4)
plt.contourf(X, Y, Z, 20)

# Improve the spacing between subplots
plt.tight_layout()

# Display the figure
plt.show()
```

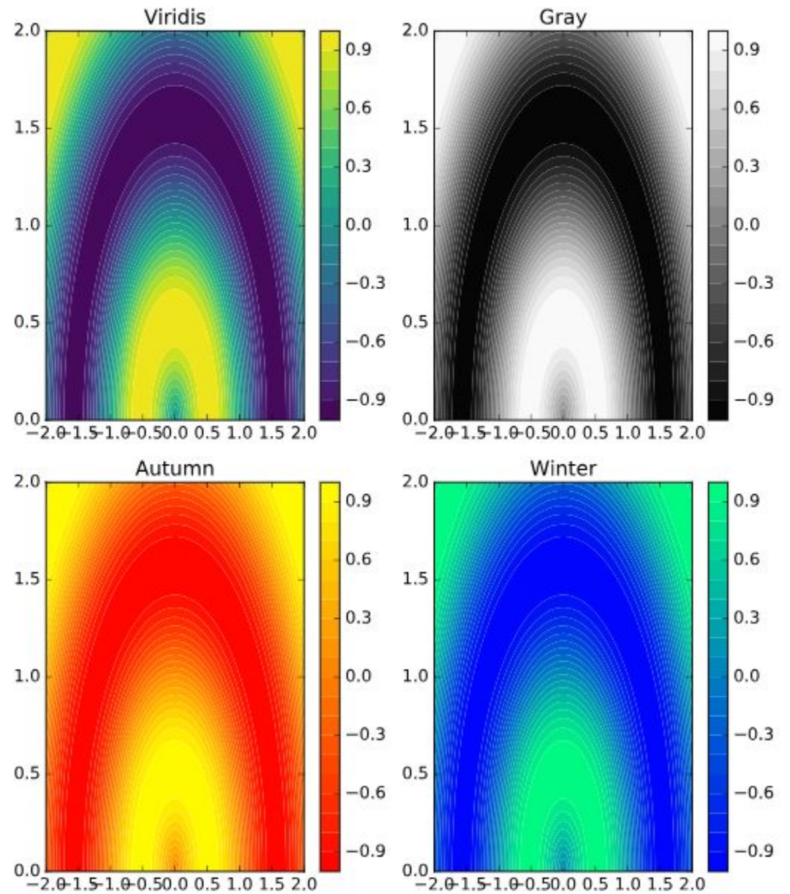


GSA

D2D
DATA TO DECISIONS

Modifying colormaps

```python
# Create a filled contour plot with a color map of 'viridis'
plt.subplot(2,2,1)
plt.contourf(X,Y,Z,20, cmap='viridis')
plt.colorbar()
plt.title('Viridis')

# Create a filled contour plot with a color map of 'gray'
plt.subplot(2,2,2)
plt.contourf(X,Y,Z,20, cmap='gray')
plt.colorbar()
plt.title('Gray')

# Create a filled contour plot with a color map of 'autumn'
plt.subplot(2,2,3)
plt.contourf(X,Y,Z,20, cmap='autumn')
plt.colorbar()
plt.title('Autumn')

# Create a filled contour plot with a color map of 'winter'
plt.subplot(2,2,4)
plt.contourf(X,Y,Z,20, cmap='winter')
plt.colorbar()
plt.title('Winter')

# Improve the spacing between subplots and display them
plt.tight_layout()
plt.show()
```

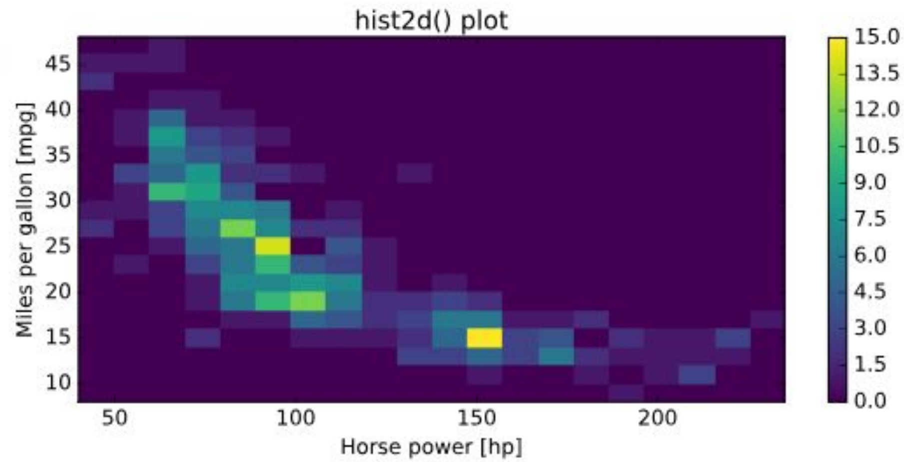


GSA

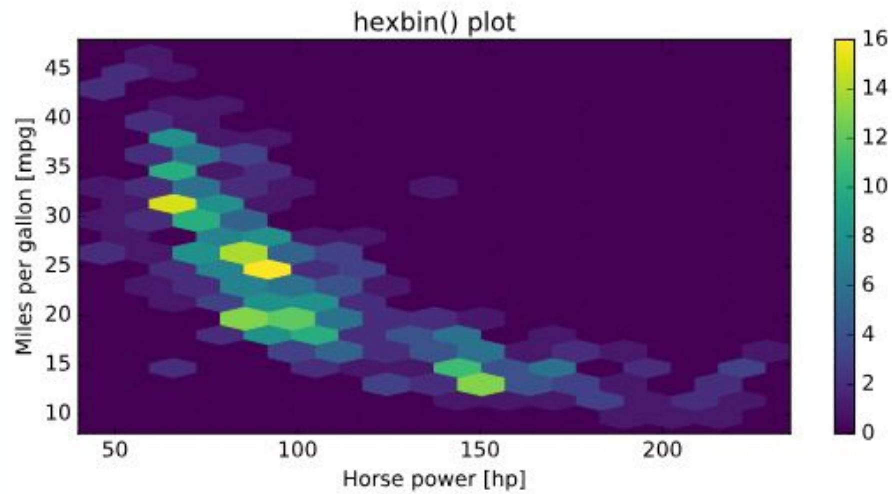
D2D
DATA TO DECISIONS

Using hist2d() and hexbin()

```
# Generate a 2-D histogram
plt.hist2d(hp, mpg, bins = (20, 20), range = ((40, 235), (8, 48)))

# Add a color bar to the histogram
plt.colorbar()

# Add labels, title, and display the plot
plt.xlabel('Horse power [hp]')
plt.ylabel('Miles per gallon [mpg]')
plt.title('hist2d() plot')
plt.show()
```




```
# Generate a 2d histogram with hexagonal bins
plt.hexbin(hp, mpg, gridsize=(15,12),
           extent=(40,235,8, 48))

# Add a color bar to the histogram
plt.colorbar()

# Add labels, title, and display the plot
plt.xlabel('Horse power [hp]')
plt.ylabel('Miles per gallon [mpg]')
plt.title('hexbin() plot')
plt.show()
```

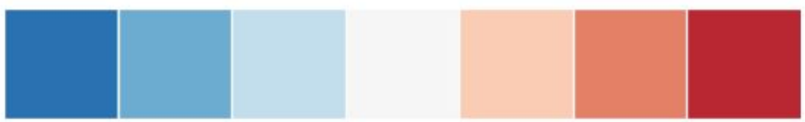


Seaborn color palettes

- https://seaborn.pydata.org/tutorial/color_palettes.html
  - ➢ Qualitative (categorical) palettes

  - ➢ Sequential palettes

  - ➢ Diverging palettes



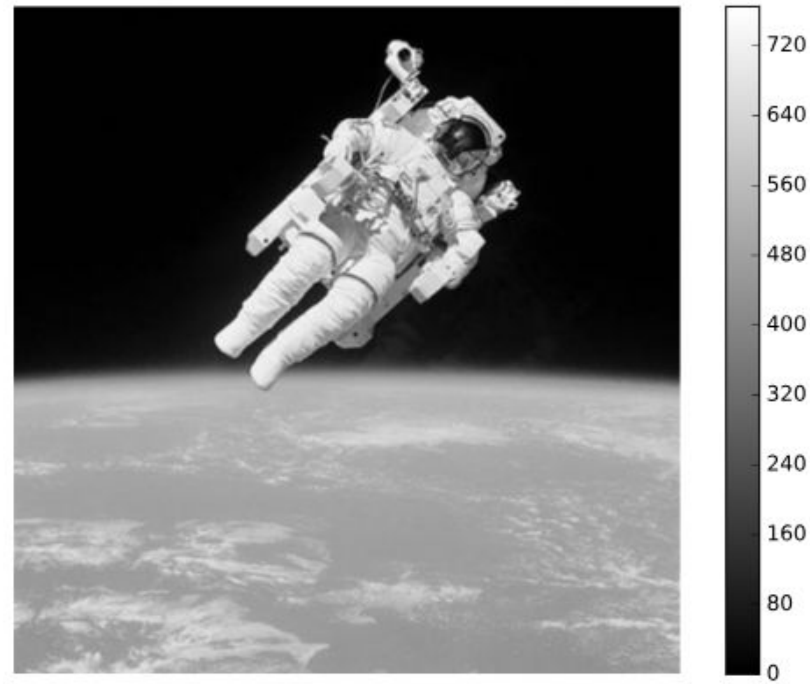
- http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3
  - ➢ Provide guidance for color blind safe design
- XKCD colors

Loading, examining images

```python
# Load the image into an array: img
img = plt.imread('480px-Astronaut-EVA.jpg')

# Print the shape of the image
print(img.shape)
(480, 480, 3)

# Display the image
plt.imshow(img)

# Hide the axes
plt.axis('off')
plt.show()

```

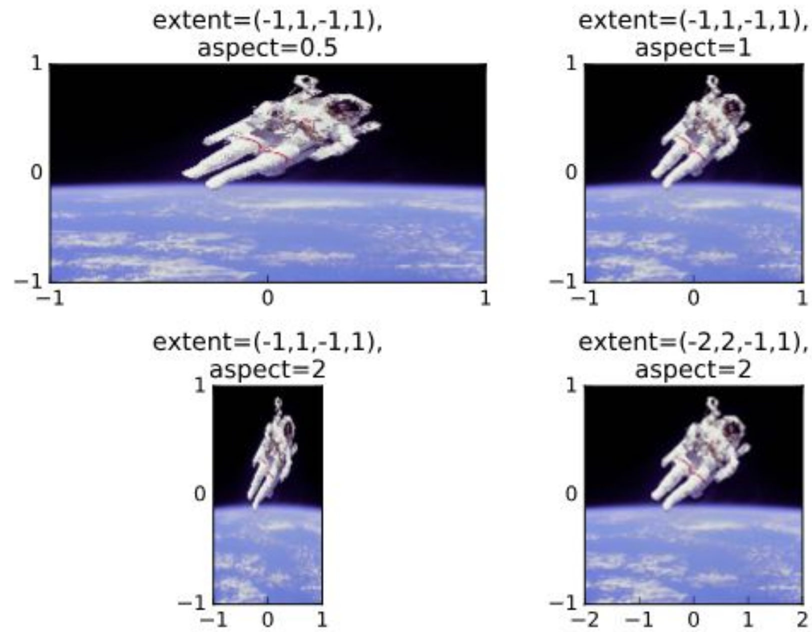
Pseudocolor plot from image data

```
# Load the image into an array: img
img = plt.imread('480px-Astronaut-EVA.jpg')

# Print the shape of the image
print(img.shape)

# Compute the sum of the red, green and blue channels: intensity
intensity = img.sum(axis = 2)

# Print the shape of the intensity
print(intensity)
(480, 480, 3)

# Display the intensity with a colormap of 'gray'
plt.imshow(intensity, cmap = 'gray')


# Add a colorbar
plt.colorbar()

# Hide the axes and show the figure
plt.axis('off')
plt.show()
```



Extent and aspect

```python
# Load the image into an array: img
img = plt.imread('480px-Astronaut-EVA.jpg')
# Specify the extent and aspect ratio of the top left subplot
plt.subplot(2,2,1)
plt.title('extent=(-1,1,-1,1),\naspect=0.5')
plt.xticks([-1,0,1])
plt.yticks([-1,0,1])
plt.imshow(img, extent = (-1,1,-1,1), aspect =0.5)
# Specify the extent and aspect ratio of the top right subplot
plt.subplot(2,2,2)
plt.title('extent=(-1,1,-1,1),\naspect=1')
plt.xticks([-1,0,1])
plt.yticks([-1,0,1])
plt.imshow(img, extent = (-1,1,-1,1), aspect =1)
# Specify the extent and aspect ratio of the bottom left subplot
plt.subplot(2,2,3)
plt.title('extent=(-1,1,-1,1),\naspect=2')
plt.xticks([-1,0,1])
plt.yticks([-1,0,1])
plt.imshow(img, extent = (-1,1,-1,1), aspect = 2)
# Specify the extent and aspect ratio of the bottom right subplot
plt.subplot(2,2,4)
plt.title('extent=(-2,2,-1,1),\naspect=2')
plt.xticks([-2,-1,0,1,2])
plt.yticks([-1,0,1])
plt.imshow(img, extent = (-2,2,-1,1), aspect = 2)
# Improve spacing and display the figure
plt.tight_layout()
plt.show()
```



Rescaling pixel intensities

```python
# Load the image into an array: image
image = plt.imread('640px-Unequalized_Hawkes_Bay_NZ.jpg')

# Extract minimum and maximum values from the image: pmin, pmax
pmin, pmax = image.min(), image.max()
print("The smallest & largest pixel intensities are %d & %d." % (pmin, pmax))

# Rescale the pixels: rescaled_image
rescaled_image = 256*(image - pmin) / (pmax - pmin)
print("The rescaled smallest & largest pixel intensities are %.1f & %.1f." %
      (rescaled_image.min(), rescaled_image.max()))

# Display the original image in the top subplot
plt.subplot(2,1,1)
plt.title('original image')
plt.axis('off')
plt.imshow(image)

# Display the rescaled image in the bottom subplot
plt.subplot(2,1,2)
plt.title('rescaled image')
plt.axis('off')
plt.imshow(rescaled_image)

plt.show()
```

original image

rescaled image

GSA

D2D DATA TO DECISIONS

Turtle

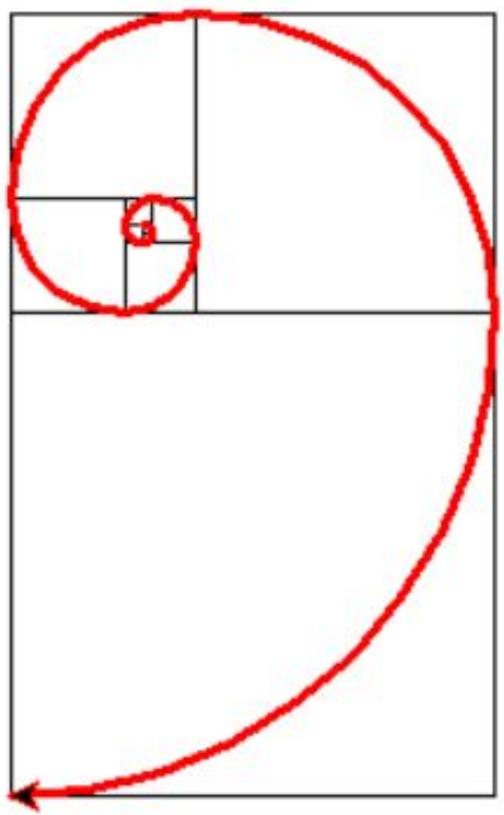
https://docs.python.org/2/library/turtle.html
http://www.cs.middlebury.edu/~mlinderman/courses/middsip/sm17/lectures/lecture4-notes.html

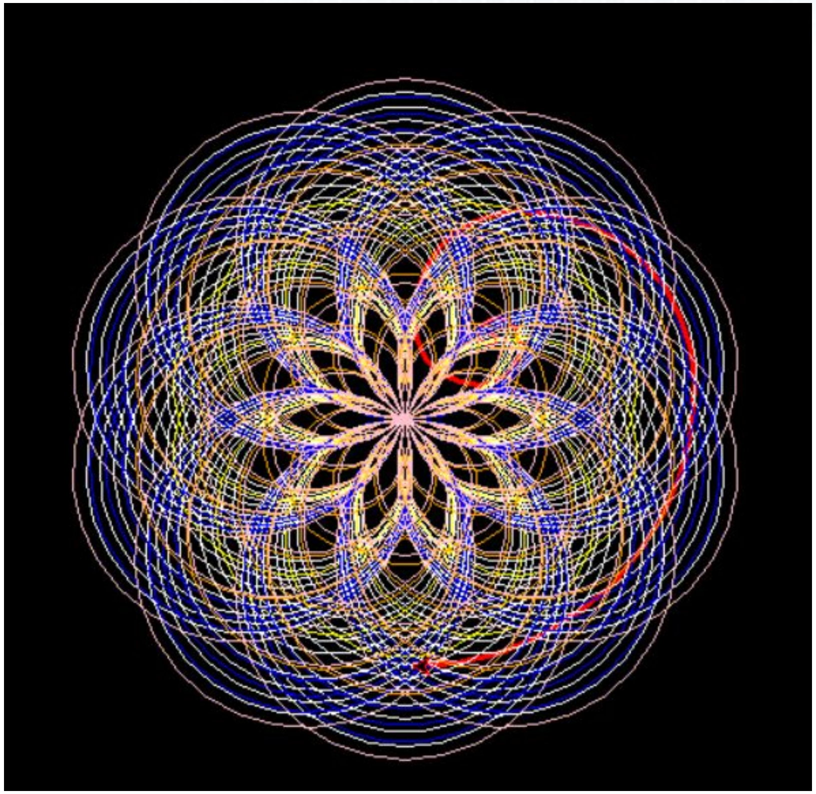
- Turtle graphics is a popular way for introducing programming to kids
- It was part of the original Logo programming language developed by Wally Feurzig and Seymour Papert in 1966
- Imagine a robotic turtle starting at (0, 0) in the x-y plane
  - Give it the command turtle.forward(15), and it moves (on-screen!) 15 pixels in the direction it is facing, drawing a line as it moves
  - Give it the command turtle.right(25), and it rotates in-place 25 degrees clockwise.
- By combining together these and similar commands, intricate shapes and pictures can easily be drawn.

GSA

D2D DATA TO DECISIONS

Mathematical drawing with Turtle





Q & A