# Lecture 4

## Data visualization with Matplotlib and Seaborn libraries

Aug 01

# Why Plot?

- For small number of values
  - 100 > 10 > 1

The Magical Number Seven Plus or Minus Two

Miller 1956

- Human “working memory” holds 7 ± 2 objects
- Most useful data is composed of >> 7 samples

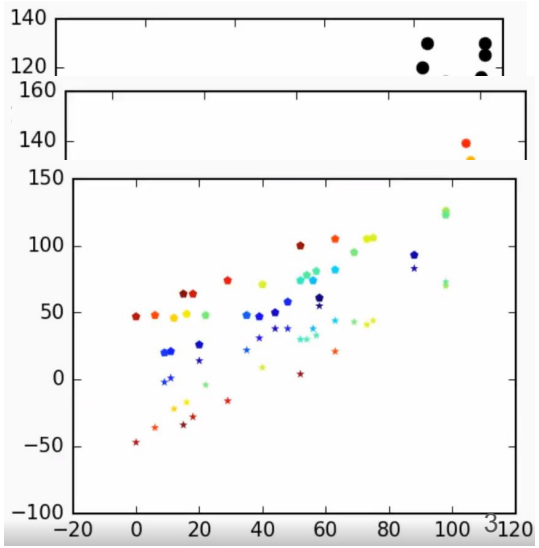
- So, for 5-9 numbers no need to plot.

# Complex Data

- Most data is not uni-dimensional.
- We usually have arrays, tensors, high dimensional data.
- Data is related to other types of data, correlation, dependence.

- However, we have only finite dimensions to visualize
  - X, Y (maybe Z)
  - Colors (visible range)
  - Shape/patterns also limited

- Aim is to convert quantitative (numbers) information to qualitative.

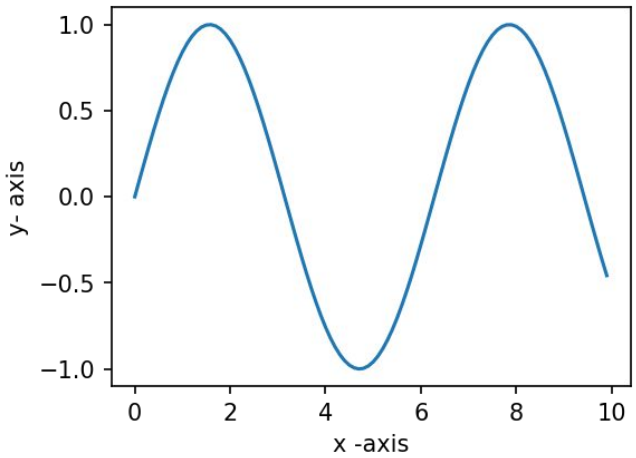


# 1. Basic Plots

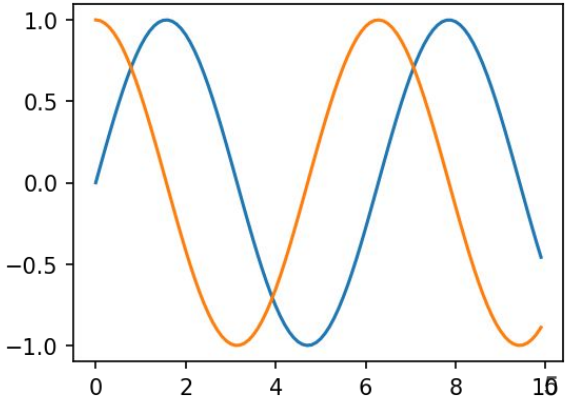
- Make 3 types of simple plots
- Changing color, line and marker style.
- Labelling the plots

# 1.1 Plotting with 2 arguments

- Line plot
  - Import matplotlib.pyplot as plt
  - plt.plot(xvals, np.sin(xvals))
  - plt.xlabel("x -axis")
  - plt.ylabel("y- axis")

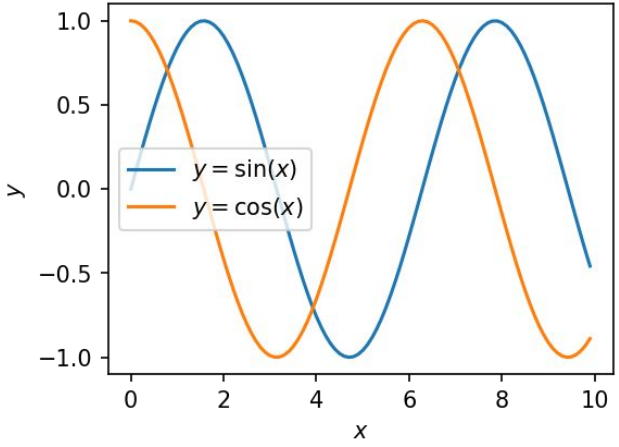



- Multiple line plot
  - plt.plot(xvals, np.sin(xvals))
  - plt.plot(xvals, np.cos(xvals))

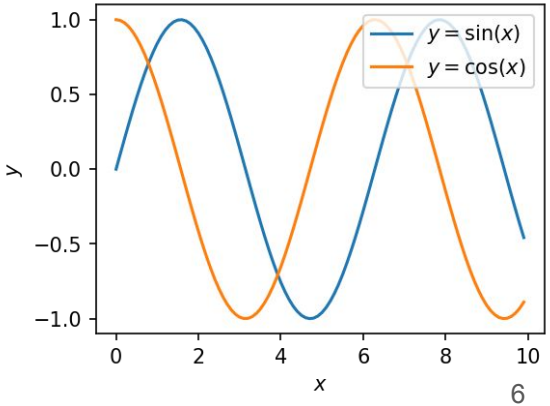


# 1.2 Using Labels

- Line plot
  - plt.plot(xvals, np.sin(xvals), label = r"$y = \sin(x)$")
  - plt.plot(xvals, np.cos(xvals), label = r"$y = \cos(x)$")
  - plt.legend()
  - plt.xlabel(r"$x$")
  - plt.ylabel(r"$y$")

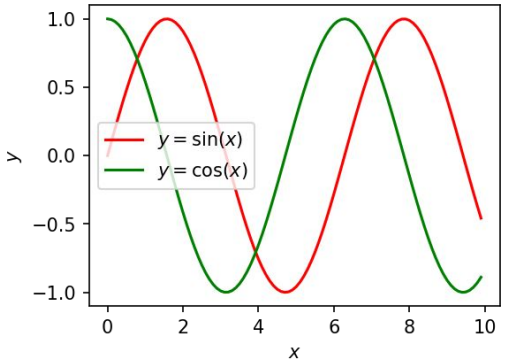



- Label position
  - plt.plot(xvals, np.sin(xvals), label = r"$y = \sin(x)$")
  - plt.plot(xvals, np.cos(xvals), label = r"$y = \cos(x)$")
  - plt.xlabel(r"$x$")
  - plt.ylabel(r"$y$")
  - plt.legend(loc= 'upper right')

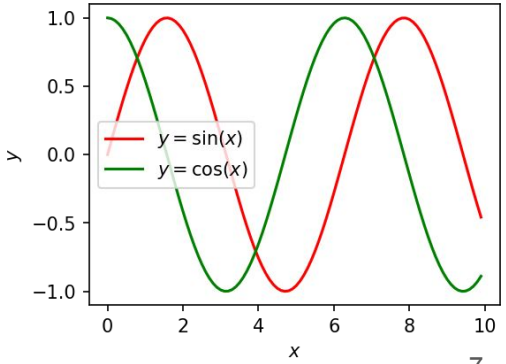


# 1.3 Changing Attributes

- Line color
  - plt.plot(xvals, np.sin(xvals), label=r"$y = \sin(x)$", color='red')
  - plt.plot(xvals, np.cos(xvals), label=r"$y = \cos(x)$", color= 'green' )
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend()

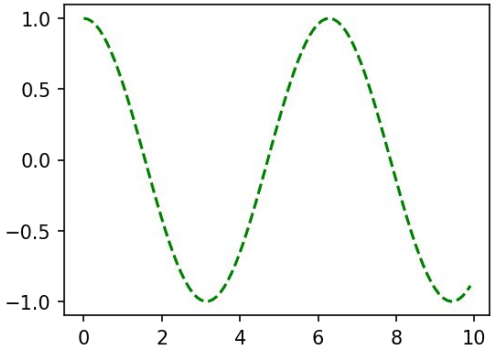



- Short keywords
  - plt.plot(xvals, np.sin(xvals), label=r"$y = \sin(x)$", color='r')
  - plt.plot(xvals, np.cos(xvals), label=r"$y = \cos(x)$", color= 'g')
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend()



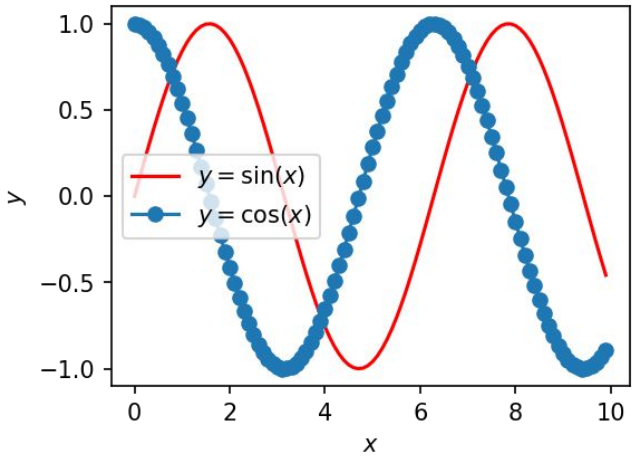
# 1.4 Changing Attributes

- Line style
  - plt.plot(xvals, np.cos(xvals), label=r"$y = \cos(x)$", color= 'green' , linestyle='--')

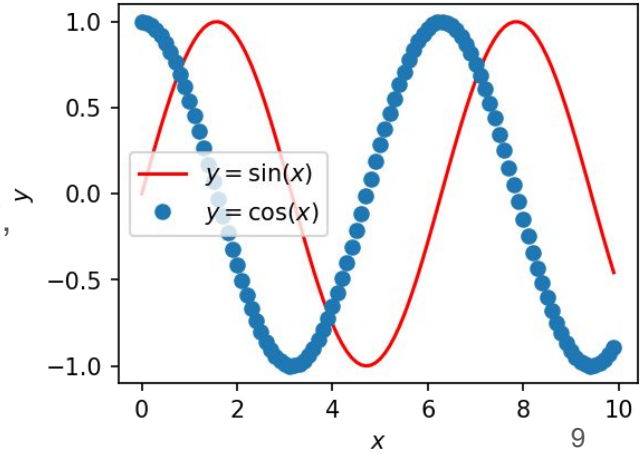


# 1.5 Introducing Markers

- Markers
  - plt.plot(xvals, np.sin(xvals), label=r"$y = \sin(x)$", color='r')
  - plt.plot(xvals, np.cos(xvals), label=r"$y = \cos(x)$", marker='o')
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend()

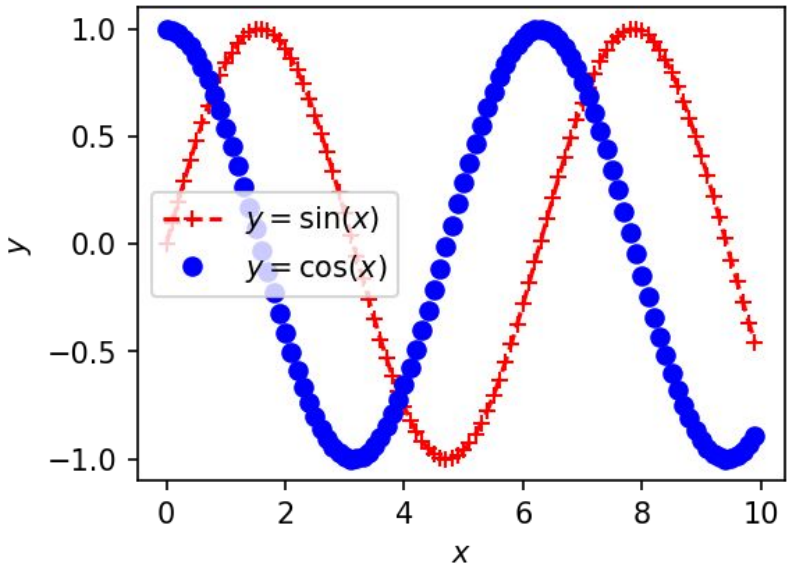



- Markers - only
  - plt.plot(xvals, np.sin(xvals), label=r"$y = \sin(x)$", color='r')
  - plt.plot(xvals, np.cos(xvals), label=r"$y = \cos(x)$", marker='o', linestyle='None')
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend()



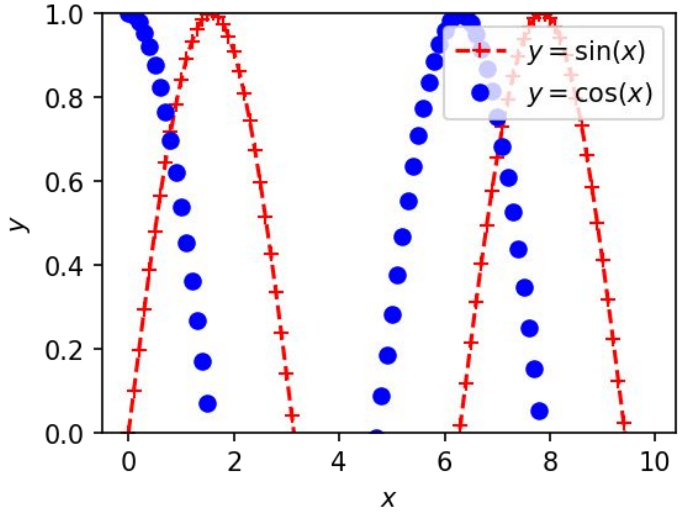
# 1.6 Short code

- Combining colorstyle, linestyle & marker style
  - plt.plot(xvals, np.sin(xvals), **'r+--'**, label=r"$y = \sin(x)$")
  - plt.plot(xvals, np.cos(xvals), **'bo'**, label=r"$y = \cos(x)$")
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend()



# 1.7 Change axis limits

- Combining colorstyle, linestyle & marker style
  - plt.plot(xvals, np.sin(xvals), 'r+--', label=r"$y = \sin(x)$")
  - plt.plot(xvals, np.cos(xvals), 'bo', label=r"$y = \cos(x)$")
  - plt.ylim(0, 1)
  - plt.xlabel("$x$")
  - plt.ylabel(r"$y$")
  - plt.legend(loc= 'upper right')

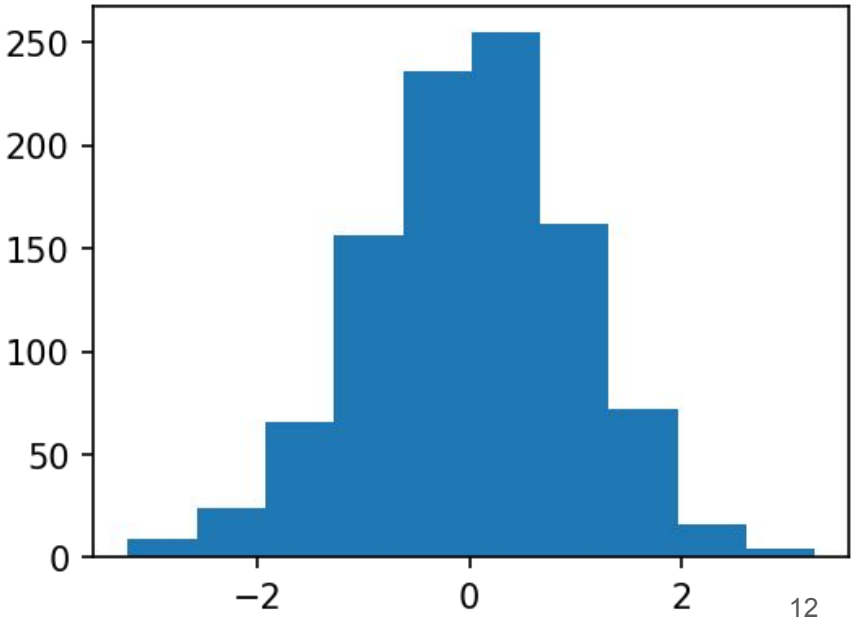


# 1.8 Histograms

- Creating a dataset
  - rands = np.random.normal(size=1000)
- Visualize it
  - plt.hist(rands)

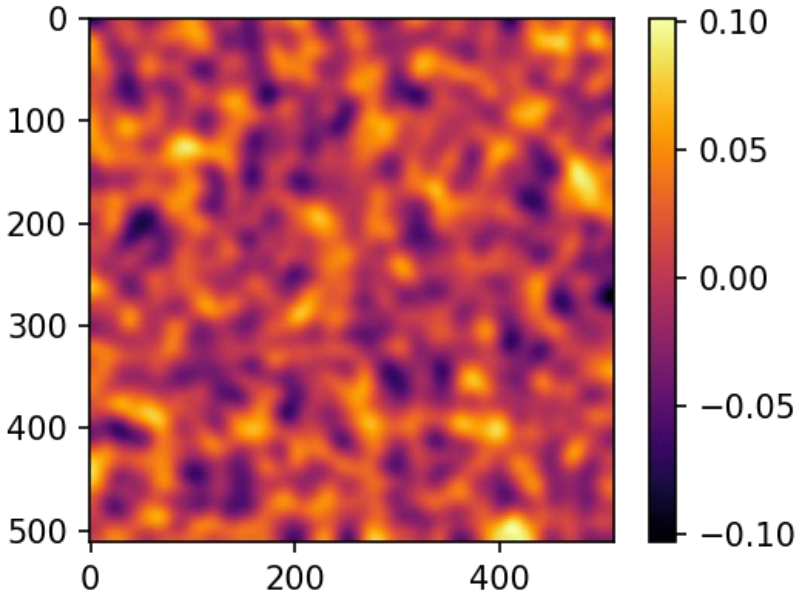
- Automatically selects bins for u
- Can change ‘bin’ properties (Later)

```
(array([  9.,  24.,  66., 156., 236., 255., 162.,
72.,  16.,   4.]),
 array([-3.21879203, -2.57102222, -1.92325241,
-1.2754826 , -0.62771278,
        0.02005703,  0.66782684,  1.31559665,
1.96336647,  2.61113628,
        3.25890609]),
 <a list of 10 Patch objects>)
```



# 1.9 3-D plots with z-axis

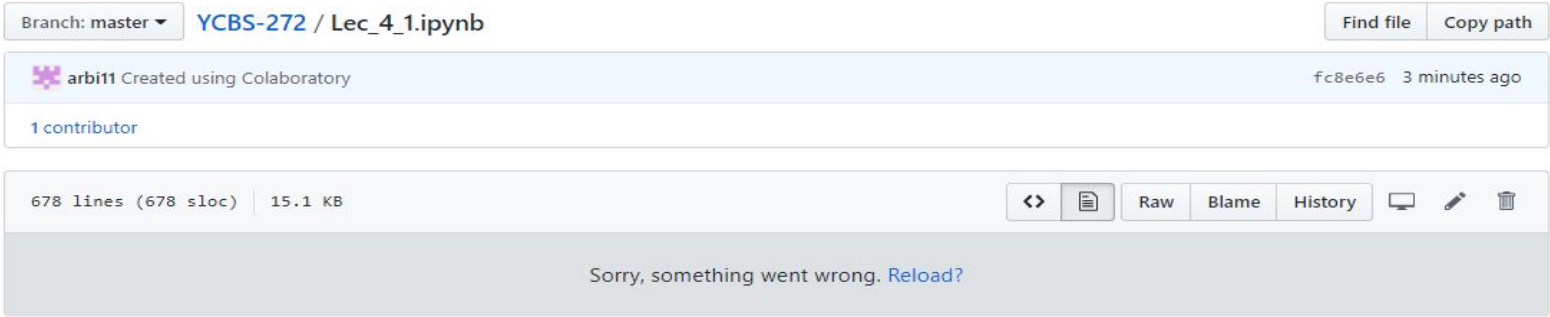
- Generate the data
  - from scipy.ndimage.filters import gaussian_filter
  - rands2d = gaussian_filter(np.random.normal(size=(512,512)), sigma=10)
  - print(rands2d.shape)
  - `(512, 512)`
- Plotting the data as an image
  - plt.imshow(rands2d, cmap='inferno')
  - plt.colorbar()



# Link for the notebook

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_1.ipynb

If you see this error on github




Copy the link (of github page) and paste here:

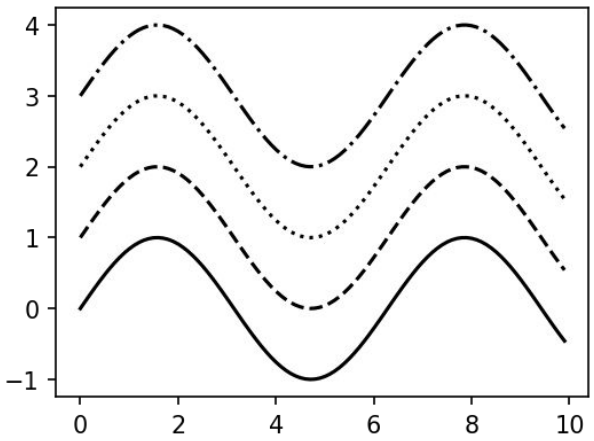
https://nbviewer.jupyter.org/

# 2. Basic Plotting Functions

- 2.1 Line & Scatter plots
- 2.2 Bar plots & Histograms
- 2.3 Images & contours

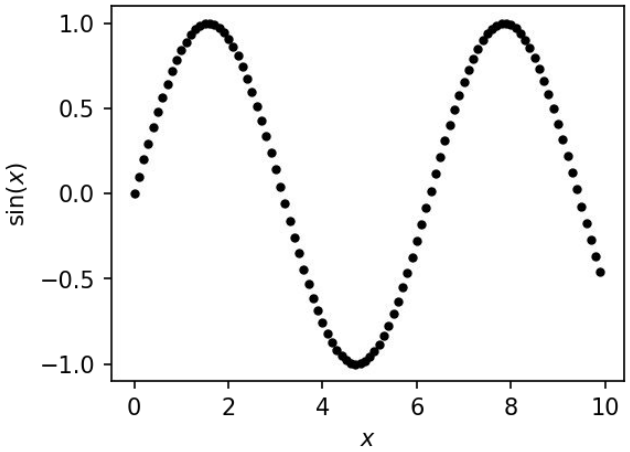
# 2.1.1 Line & Scatter Plots

- Load data
  - xvals = np.arange(0,10,0.1)
- Different line styles
  - idx = 0
  - for marker in ('-', '--', ':', '-.'):
  - plt.plot(xvals, np.sin(xvals)+idx, ''.join(('k',marker)))
  - idx += 1

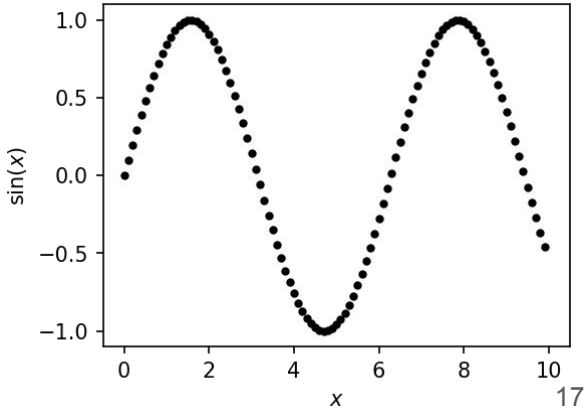


# 2.1.2 Line plots with markers

- Markers
  - plt.plot(xvals, np.sin(xvals), 'k.')
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')

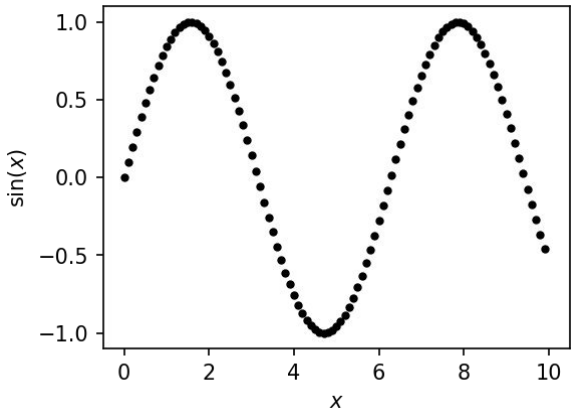



- Markers - dense
  - #Marker frequency (markevery = 10)
  - xvals2 = np.arange(0,10,0.01)
  - plt.plot(xvals2, np.sin(xvals2), 'k.', markevery=10)
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')

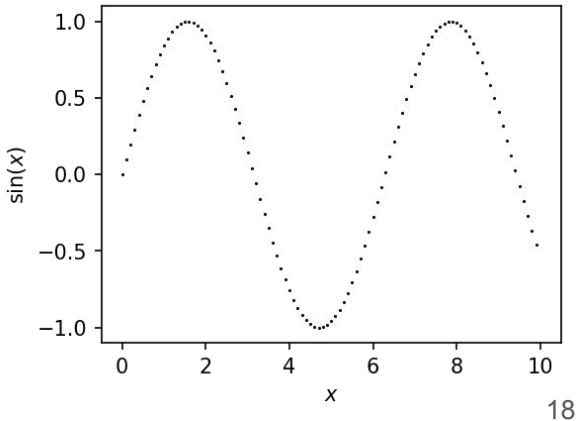


# 2.1.3 Marker Frequency

- Markevery
  - #Marker frequency (markevery = 10)
  - xvals2 = np.arange(0,10,0.01)
  - plt.plot(xvals2, np.sin(xvals2), 'k.', markevery=10)
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



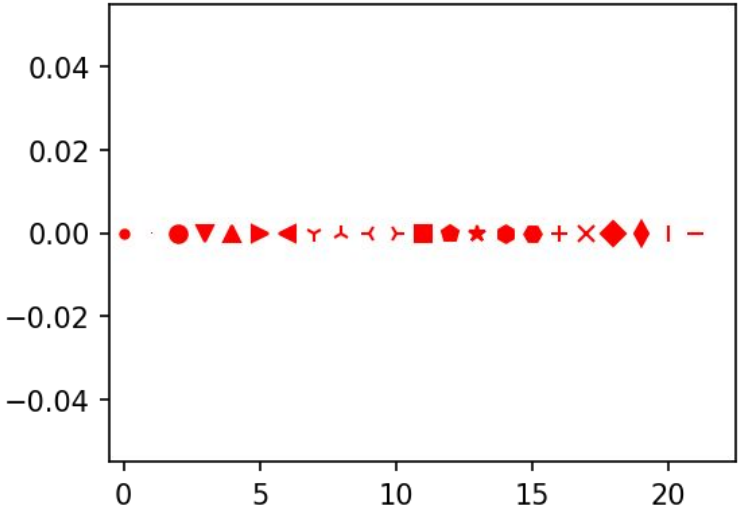

- Markersize (ms) - 1
  - xvals2 = np.arange(0,10,0.01)
  - plt.plot(xvals2, np.sin(xvals2), 'k.', markevery=10, ms= 1)
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



# 2.1.4 Wide variety of markets

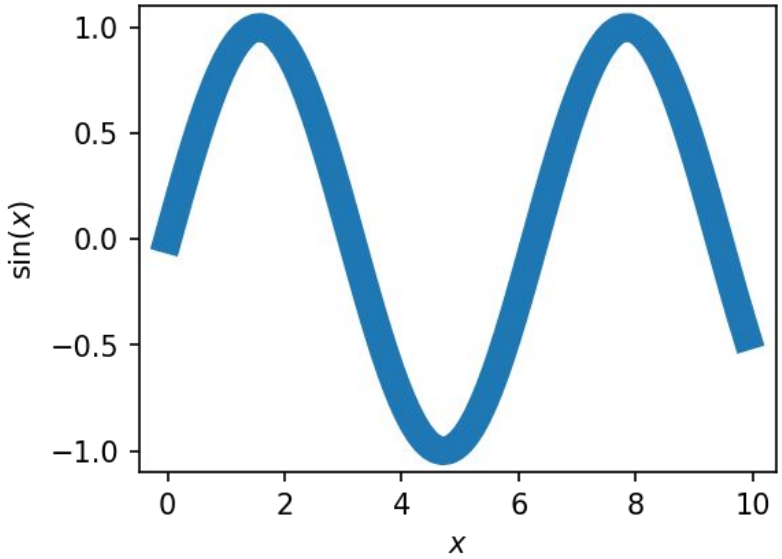
- Popular markers available

```
#different markers
idx = 0
for marker in ('.', ',', 'o', 'v', '^', '>', '<', '1', '2', '3', '4', 's', 'p',
               '*', 'h', 'H', '+', 'x', 'D', 'd', '|', '_'):
    plt.plot(idx, 0, ''.join(('r',marker)))
    idx += 1
plt.xlim(-0.5,idx+0.5)
```



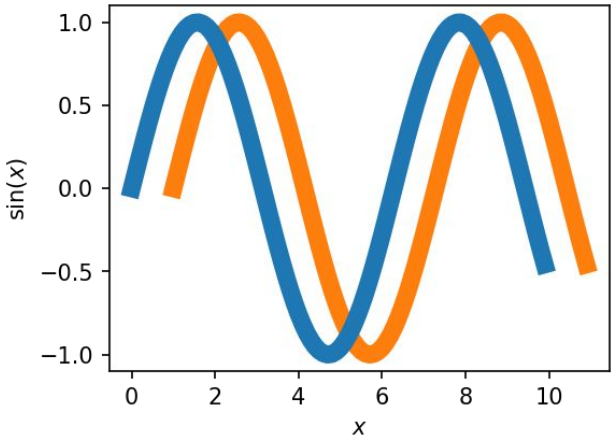
# 2.1.5 Line width

- Line widths
  - plt.plot(xvals, np.sin(xvals), linewidth=10)
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



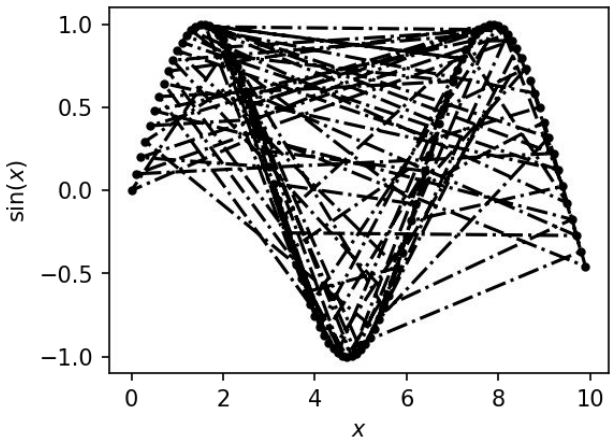
# 2.1.6 Z- order

- Order of printing
  - plt.plot(xvals, np.sin(xvals), lw=8, zorder=10)
  - plt.plot(xvals+1, np.sin(xvals), lw=8, zorder=1)
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



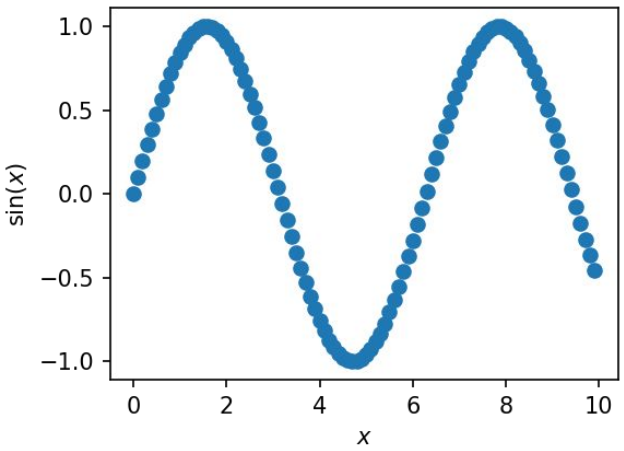
# 2.1.7 Ordered Sequence

- Array order matters!
  - shuffled_xvals = np.random.permutation(xvals)
  - plt.plot(shuffled_xvals, np.sin(shuffled_xvals), 'k.-.')
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



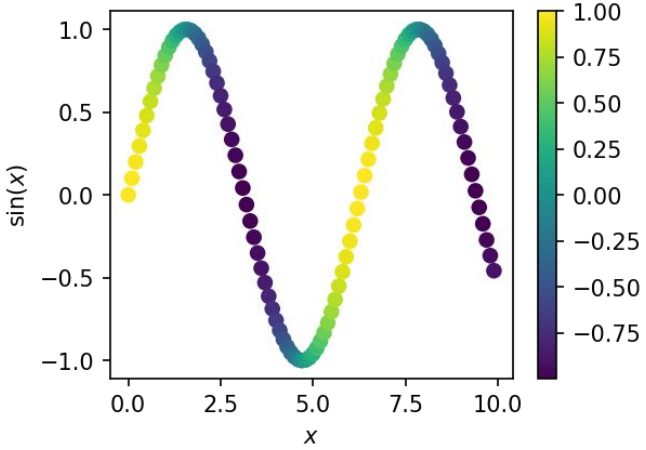
# 2.1.8 Scatter Plots

- Basic plot
  - xvals = np.arange(0,10,0.1)
  - plt.scatter(xvals, np.sin(xvals))
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



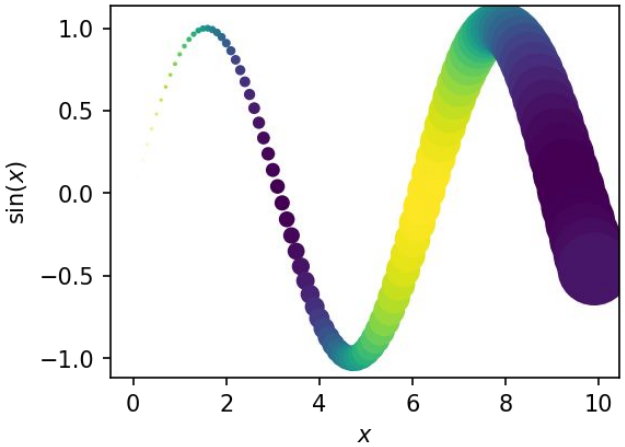
# 2.1.9 Colors

- Basic plot
  - plt.scatter(xvals, np.sin(xvals), c= np.cos(xvals))
  - plt.colorbar()
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



# 2.1.10 SIze

- Size of the marker
  - plt.scatter(xvals, np.sin(xvals), c=np.cos(xvals), s= np.power(xvals, 3))
  - plt.xlabel(r'$x$')
  - plt.ylabel(r'$\sin(x)$')



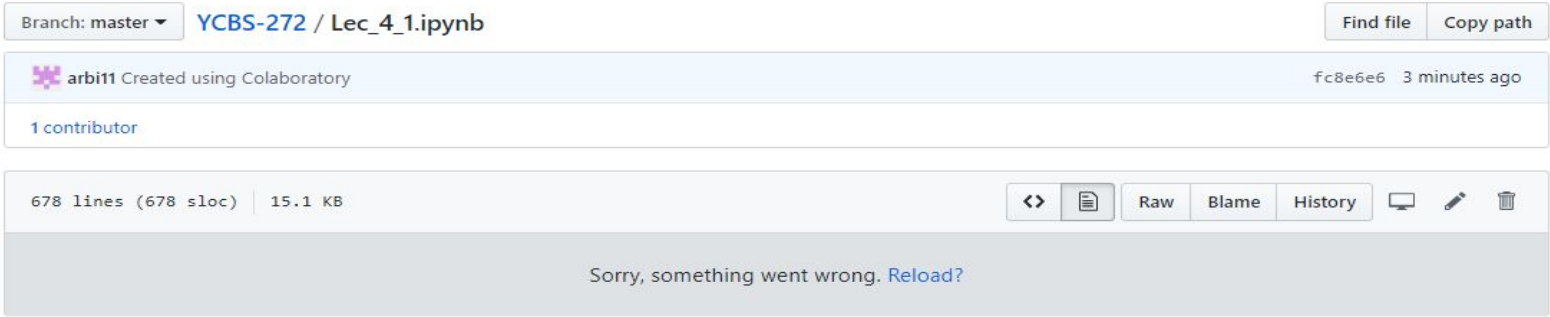
# 2.1.11 Why not use scatter always?

- Computation time!

# Link for the notebook

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_2.ipynb

If you see this error on github

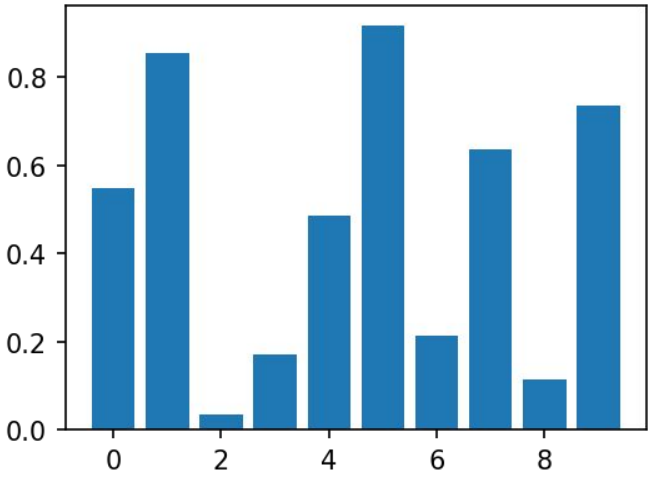



Copy the link (of github page) and paste here:

https://nbviewer.jupyter.org/

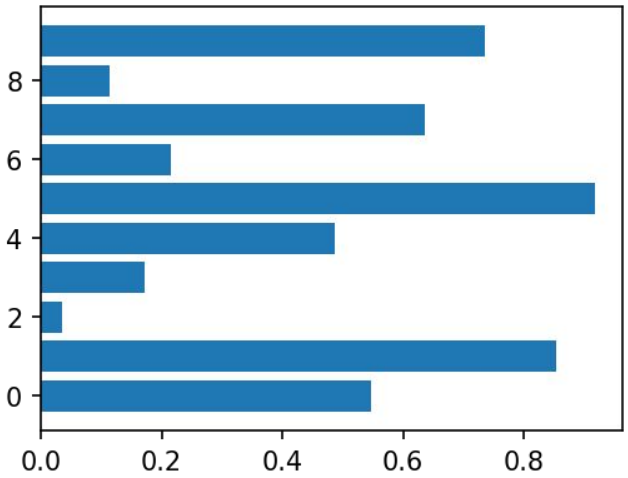
# 2.2 Bar and Histograms

- Basic bar plot
  - nums = np.random.uniform(size=10)
  - plt.bar(np.arange(10), height = nums)



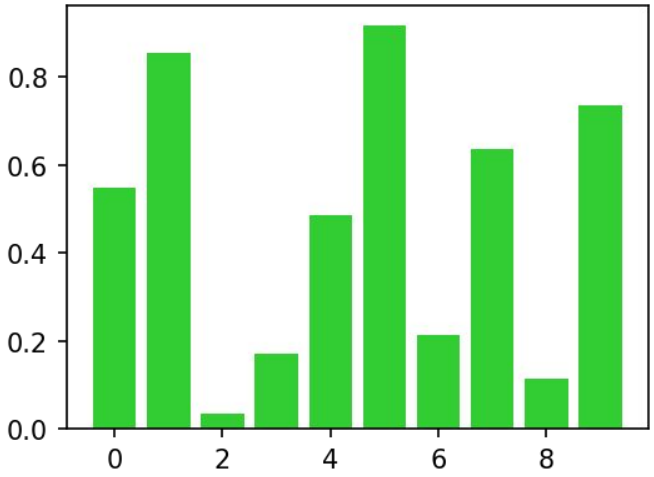
# 2.2.1 Horizontal bar plots

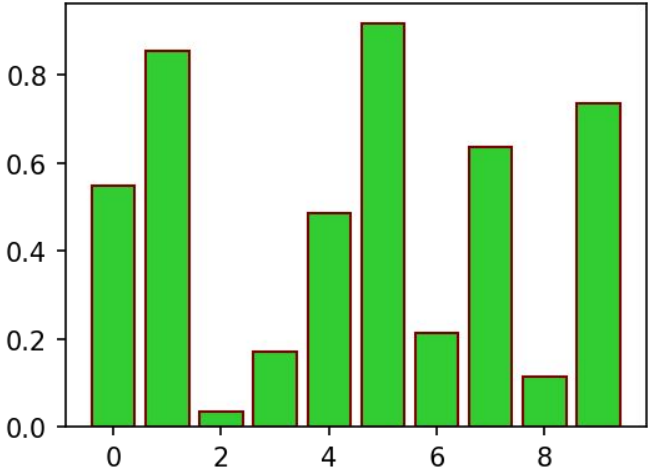
- Horizontal bar plot
  - plt.barh(np.arange(10), width = nums)



# 2.2.2 Colors

- Changing color
  - plt.bar(np.arange(10), height = nums, color='limegreen')

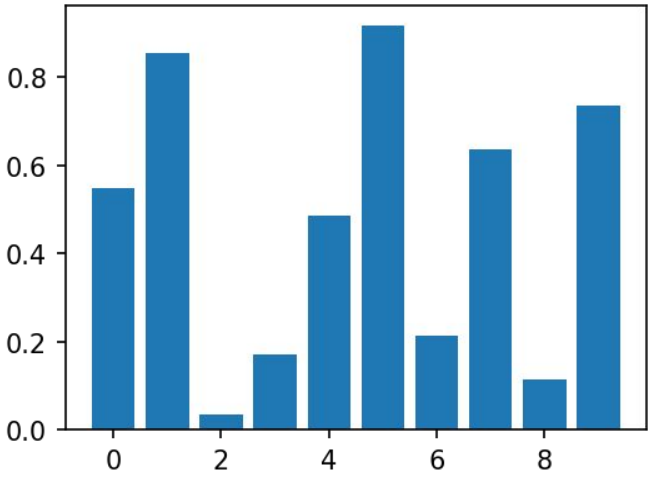



- Changing edge color
  - plt.bar(np.arange(10), height = nums, color= 'limegreen', edgecolor= 'maroon')

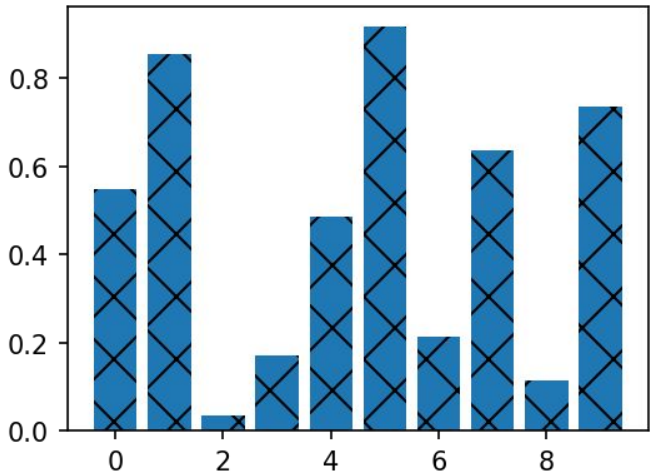
# 2.2.3 Centering

- Bar alignment
  - plt.bar(np.arange(10), nums, align='center')



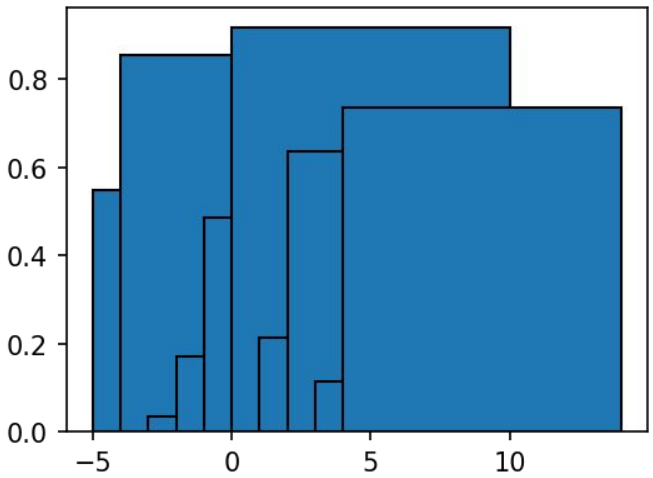
# 2.2.4 Area in a bar

- Hatch & fill
  - plt.bar(np.arange(10), nums, hatch= 'x')



# 2.2.5 Bar Width

- Width
  - plt.bar(np.arange(10), nums, width=10, edgecolor= 'k')
- Default width is 1
- Leaves 10% area before and after each bar

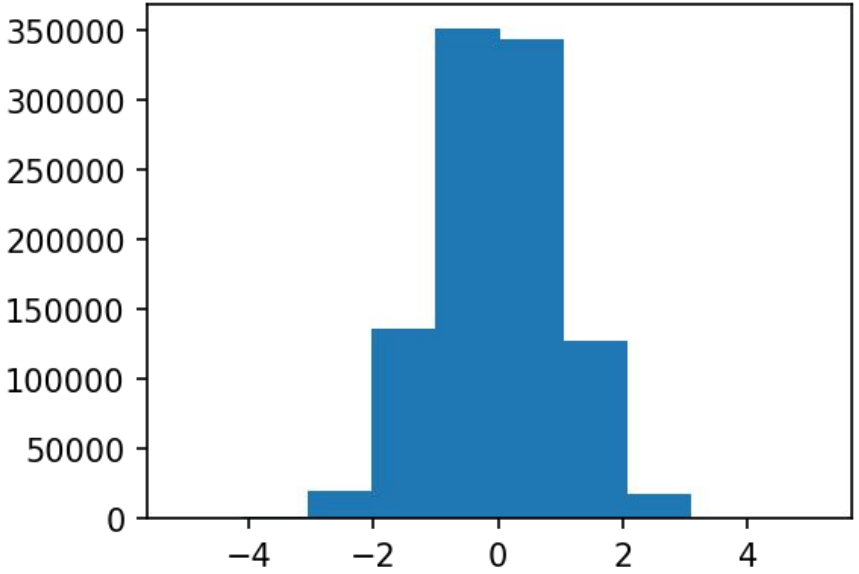


# 2.2.6 Histograms

- For Continuous data
- To plot pdf
  - # basic histogram
  - rands = np.random.normal(size=int(1e6))
- plt.hist(rands)

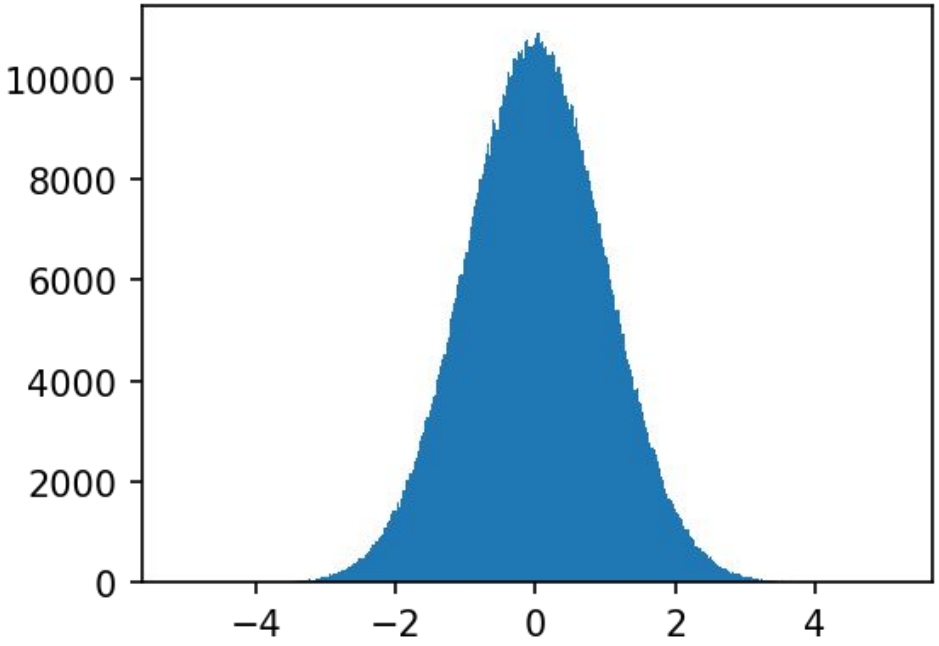
- TUPLE
  - 1st array : # elements in a bin
  - 2nd array : bin edges

```
(array([2.40000e+01, 1.04900e+03, 2.00280e+04,
1.36495e+05, 3.51514e+05,
        3.43763e+05, 1.28080e+05, 1.80450e+04,
9.85000e+02, 1.70000e+01]),
 array([-5.11976088, -4.09155511, -3.06334935,
-2.03514358, -1.00693782,
         0.02126794,  1.04947371,  2.07767947,
3.10588523,  4.134091  ,
         5.16229676]),
 <a list of 10 Patch objects>)
```



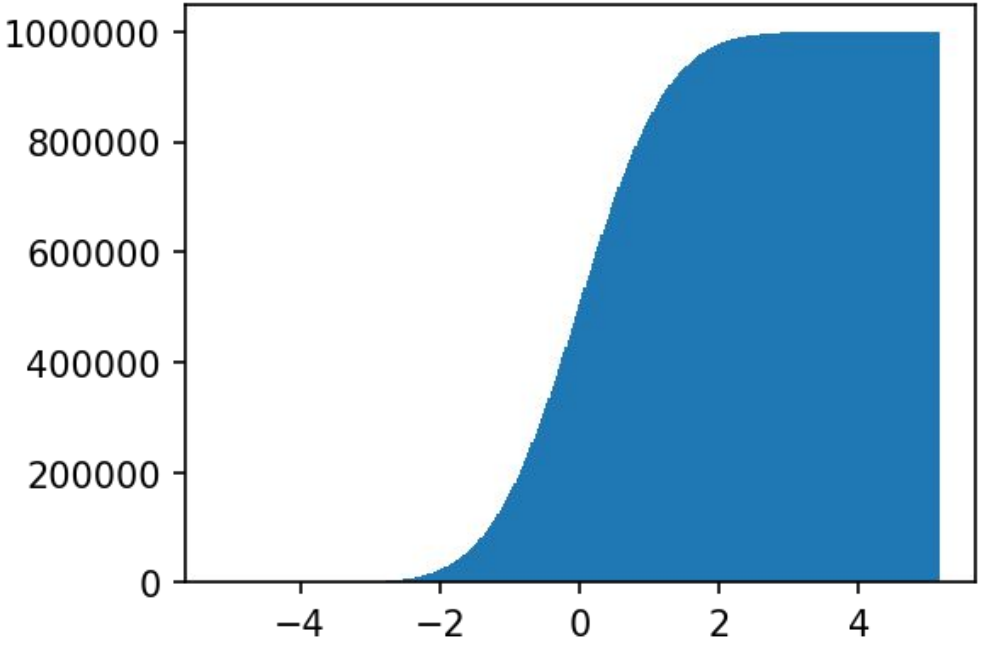
# 2.2.7 Auto binning

- plt.hist(rands, bins= 'auto')



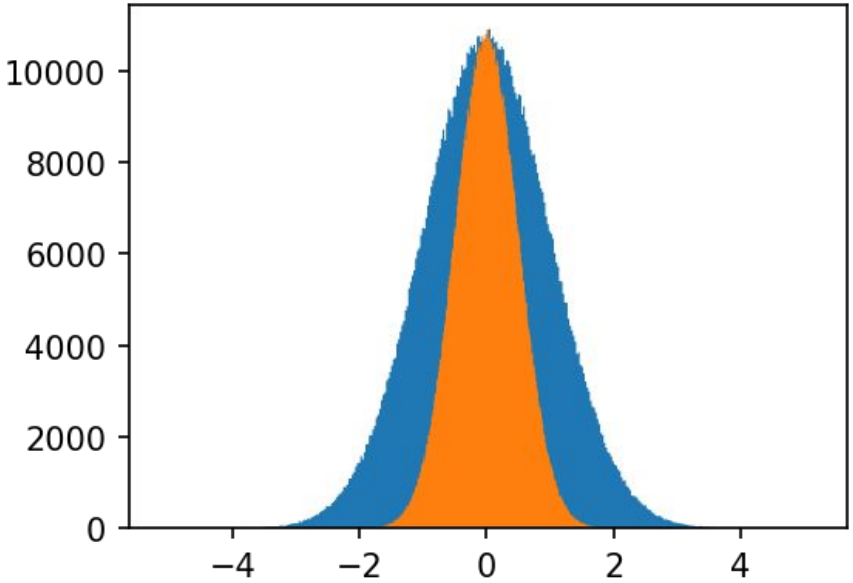
# 2.2.8 Cumulative

- CDF
  - plt.hist(rands, bins= 'auto', cumulative=True);



# 2.2.9 Multiple hist

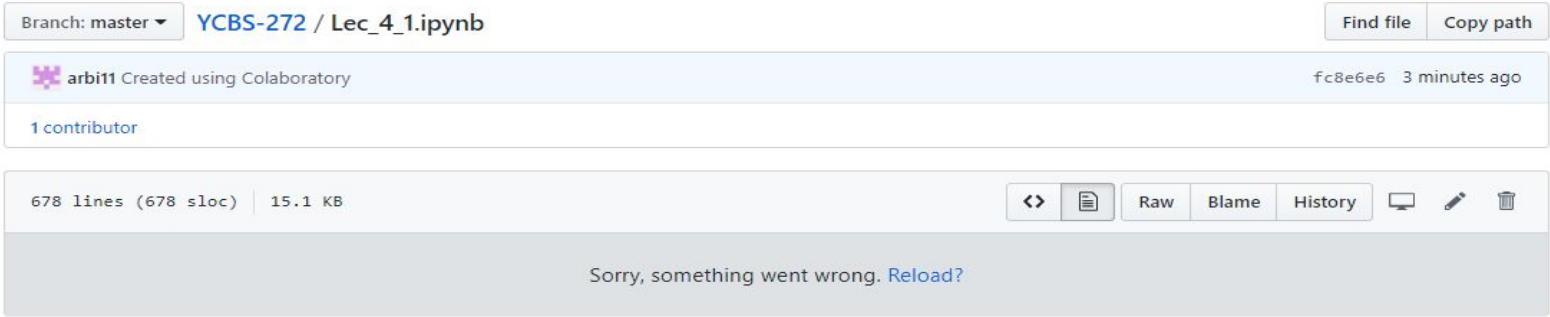
- CDF
  - plt.hist(rands, bins = 'auto', histtype= 'stepfilled');
  - plt.hist(0.5*rands, bins = 'auto', histtype= 'stepfilled');
  -



# Link for the notebook

If you see this error on github

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_3.ipynb



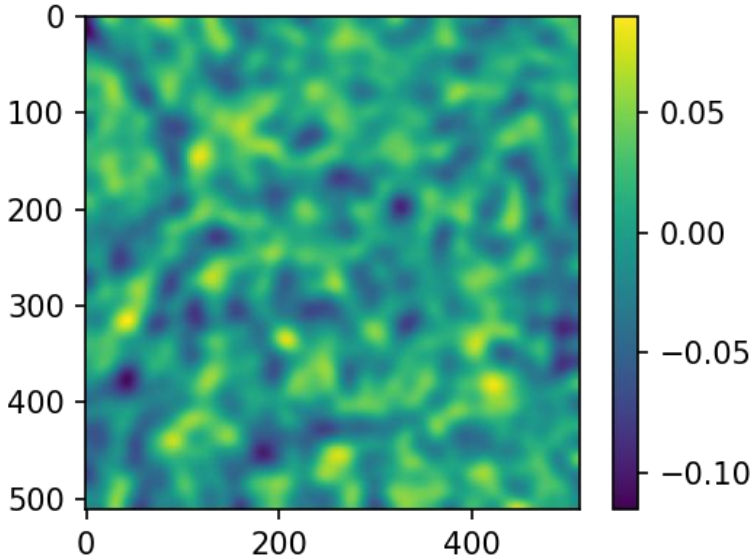

https://nbviewer.jupyter.org/

# 2.3 Images & Contours

- Making image plots with imshow()
- Tweaking images
- Using contours to highlight important regions in the data
- Combining two types of plots

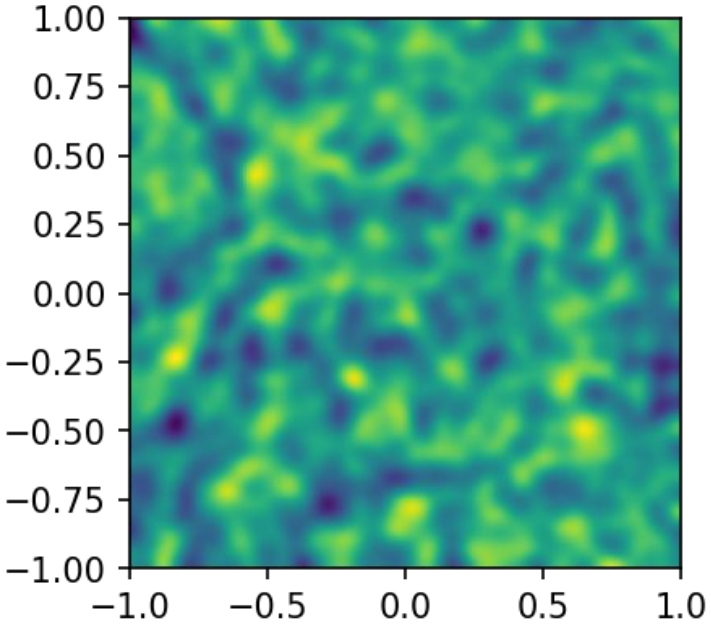
# 2.3.1 Basic plot

- Load data
  - #Generate a smoothed, gaussian random field
  - from scipy.ndimage.filters import gaussian_filter
  - rands2d = gaussian_filter(np.random.normal(size=(512,512)), sigma=10)
- Plot data
  - #Basic imshow
  - plt.imshow(rands2d)
  - plt.colorbar()



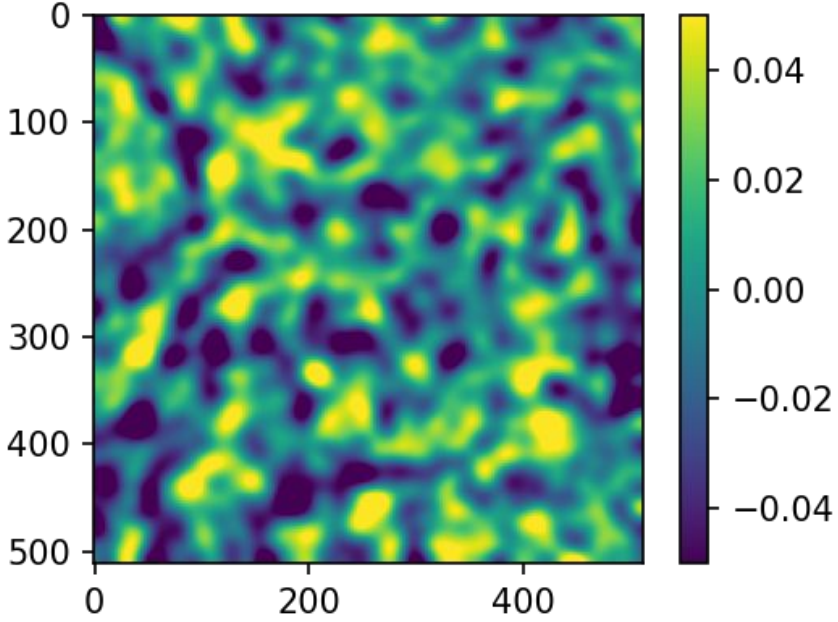
# 2.3.2 LRBT

- Extent
  - #Extent
  - plt.imshow(rands2d, extent= [-1, 1, -1, 1])



# 2.3.3 Max min of the colorbar

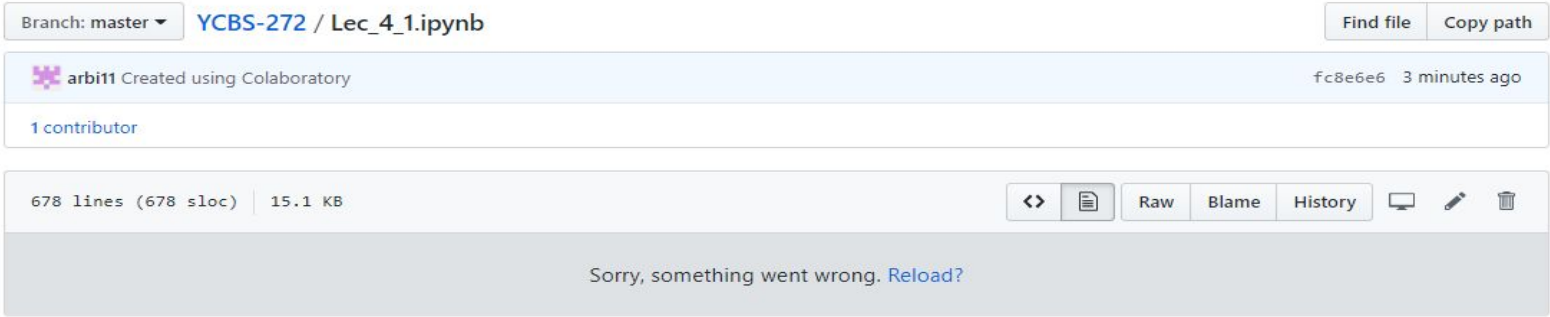
- Extent
  - #Min/Max
  - plt.imshow(rands2d, vmax= 0.05, vmin= -0.05)
  - plt.colorbar()



# Link for the notebook

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_4.ipynb

If you see this error on github



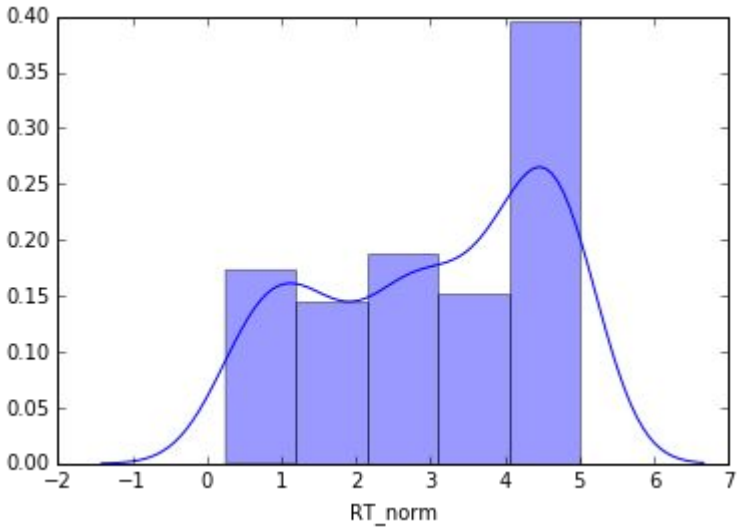

Copy the link (of github page) and paste here:

https://nbviewer.jupyter.org/

# Seaborn

- Theming Seaborn
- Statistical plots with seaborn
- Automatic generation tools

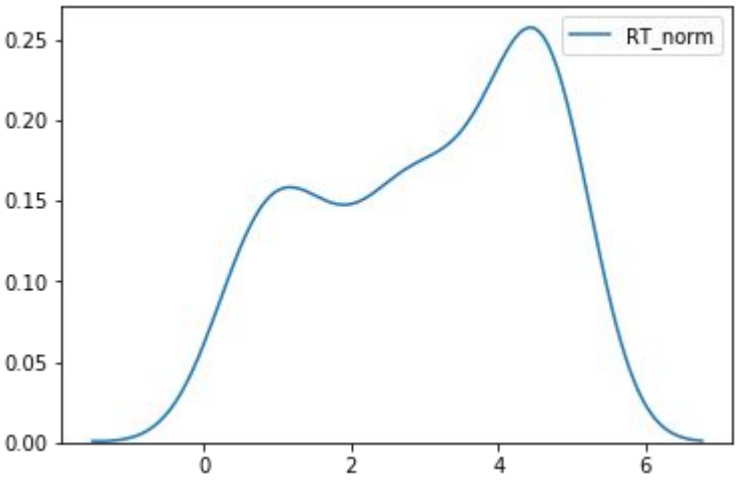
```
#Distribution plot

sns.distplot(data['RT_norm'])
```



#Distribution plot

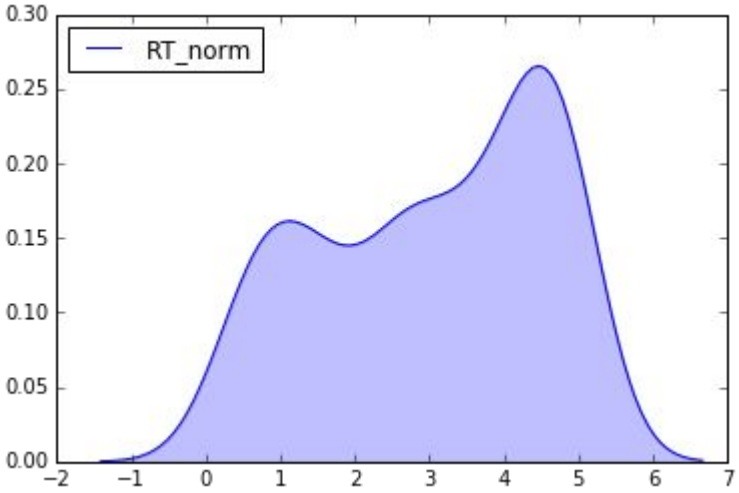
#KDE Plot

sns.kdeplot(data['RT_norm'], shade=True)

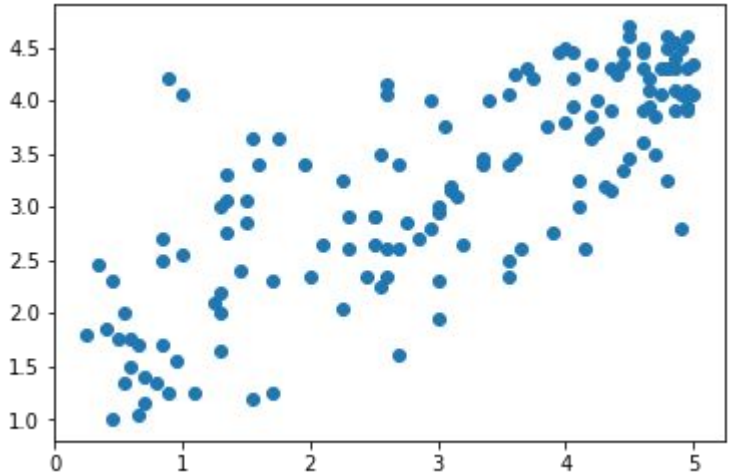


#KDE Plot

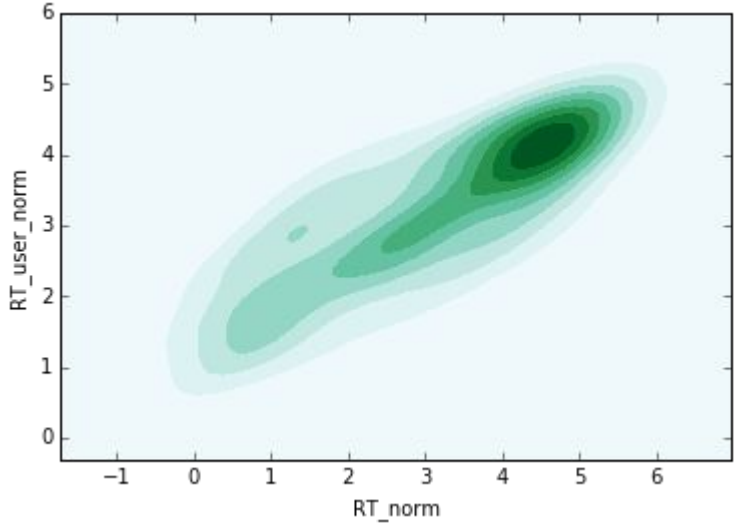
sns.kdeplot(data['RT_norm'], shade=True)



```
plt.scatter(data['RT_norm'],
data['RT_user_norm'])
```

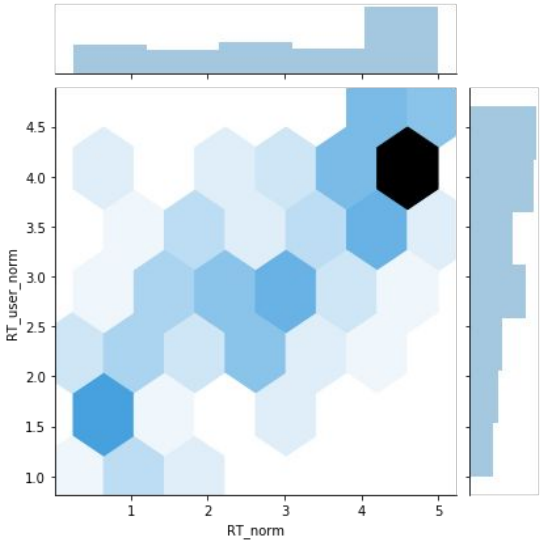



```
#2D KDE Plot

sns.kdeplot(data['RT_norm'],
data['RT_user_norm'], shade=True)
```



#The Joint plot

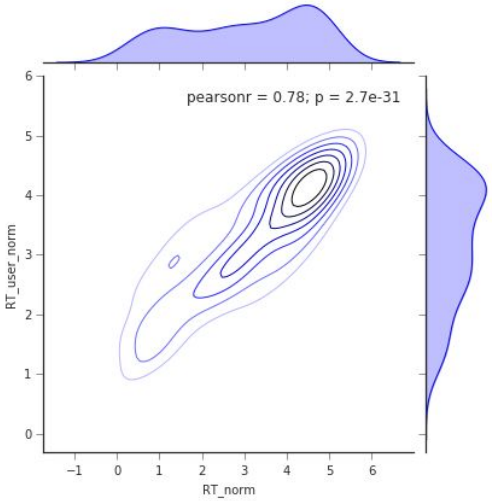
sns.jointplot(x='RT_norm', y='RT_user_norm', data=data, kind='hex')



#Theming Seaborn

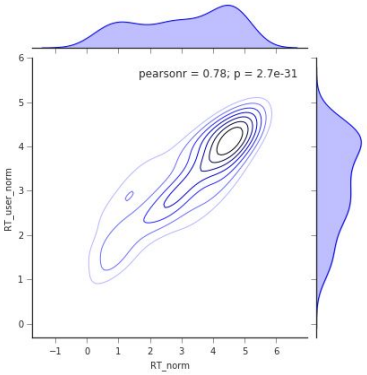
sns.set_style('ticks')

sns.jointplot(x='RT_norm', y='RT_user_norm', data=data, kind='kde', shade=False)



#Theming Seaborn

```
sns.set_style('ticks')

sns.jointplot(x='RT_norm', y='RT_user_norm', data=data, kind='kde',
shade=False)
```

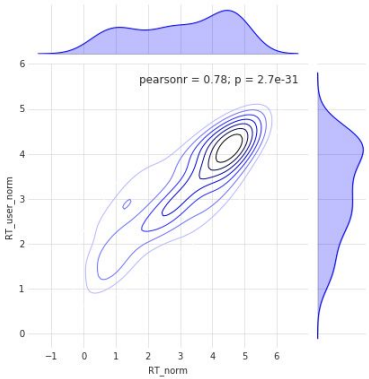


#Spine Control

sns.set_style('whitegrid')

sns.jointplot(x='RT_norm', y='RT_user_norm', data=data, kind='kde', shade=False)

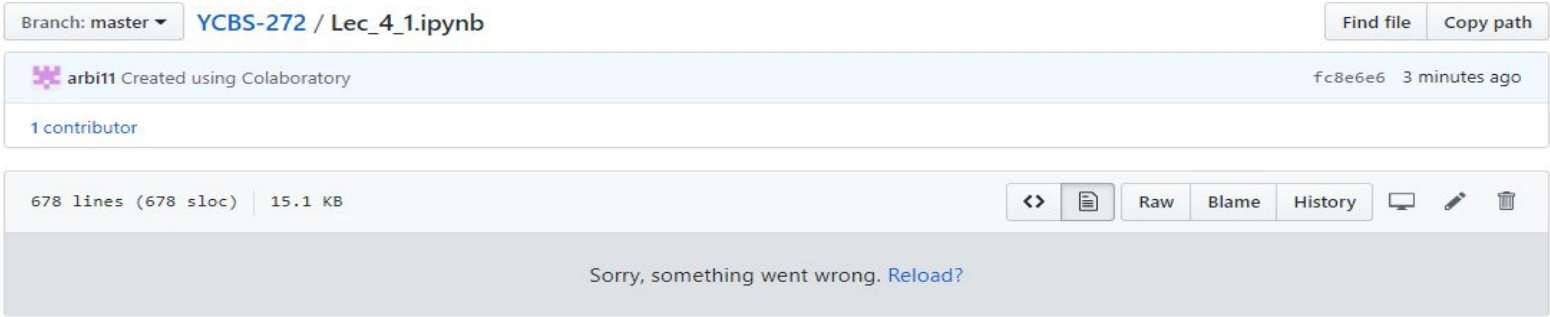
sns.despine(left=True, bottom=True)



# Link for the notebook

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_5.ipynb

If you see this error on github



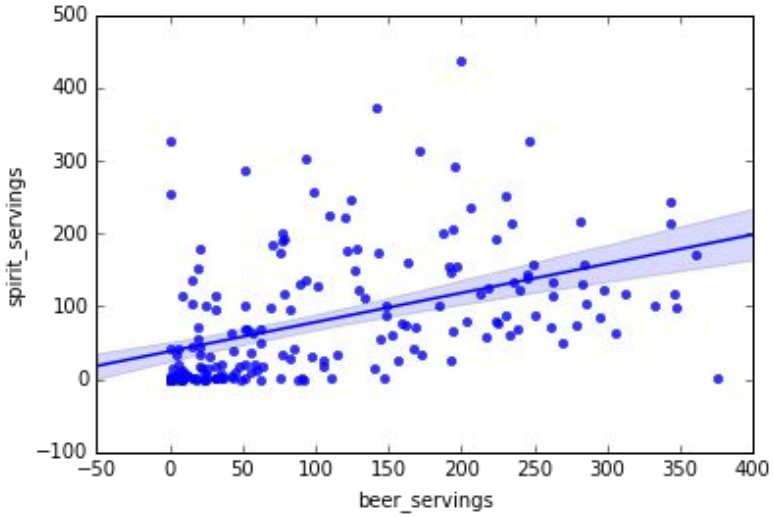

Copy the link (of github page) and paste here:

https://nbviewer.jupyter.org/

```
drinks = data.pivot_table(index="country")

#Heatmap

sns.heatmap(drinks.drop('total_litres_of_pure_alcohol', axis=1).head(10),
cbar_kws=dict(label='# of drinks'))
```

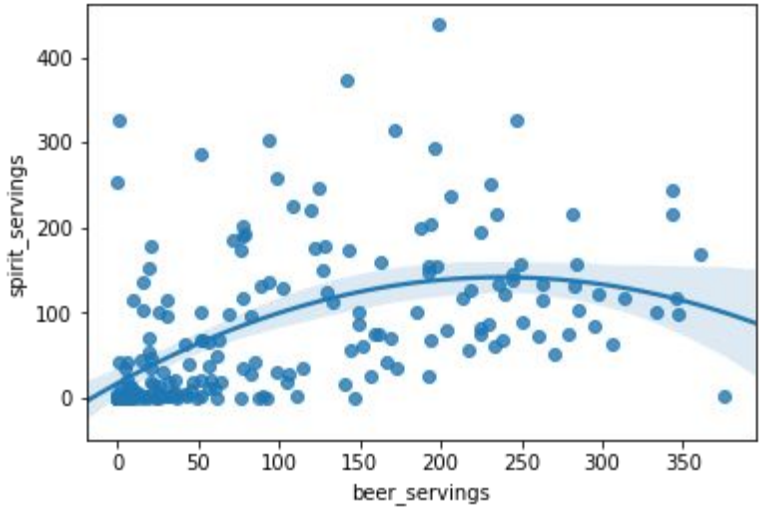
#Regression plot

sns.regplot(x='beer_servings', y='spirit_servings', data=drinks)



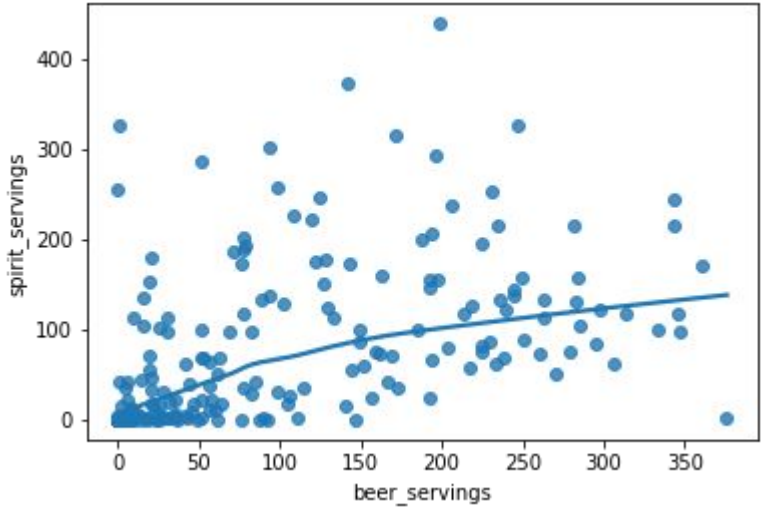
#Polynomial Regression

sns.regplot(x='beer_servings', y='spirit_servings', data=drinks, order=2)



#Lowess (LOcally Weighted linear regrESSion) (requires statsmodels!)

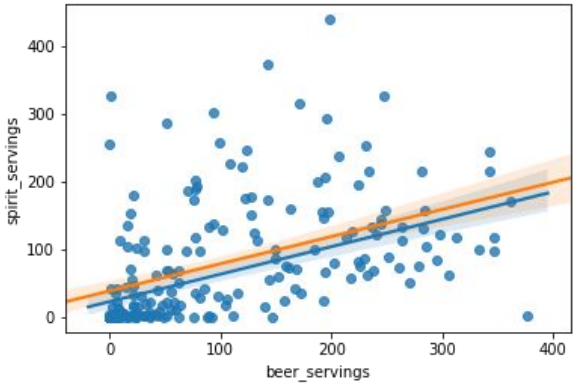
```
sns.regplot(x='beer_servings', y='spirit_servings', data=drinks, lowess=True)
```



#Robust regression

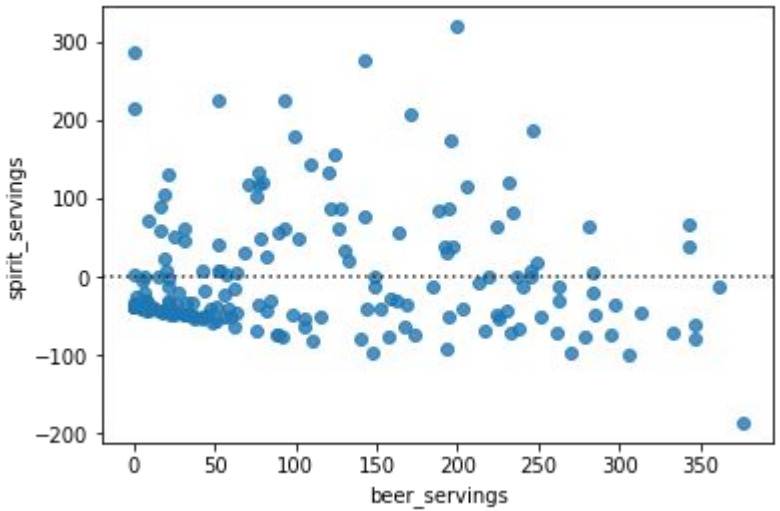
sns.regplot(x='beer_servings', y='spirit_servings', data=drinks, robust=True)

sns.regplot(x='beer_servings', y='spirit_servings', data=drinks, scatter=False, robust=False)



#Plot Residuals

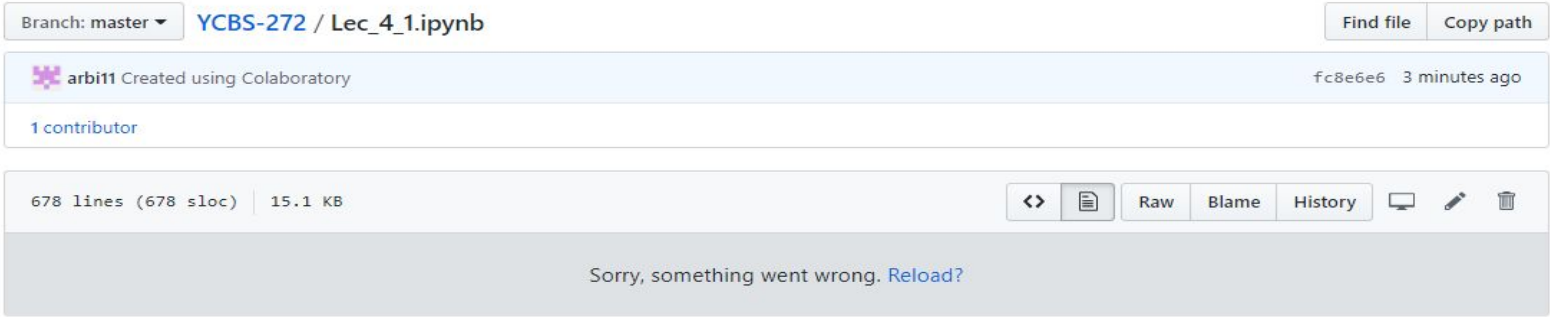
sns.residplot(x='beer_servings', y='spirit_servings', data=drinks)



# Link for the notebook

https://github.com/arbi11/YCBS-272/blob/master/Lec_4_6.ipynb

If you see this error on github




Copy the link (of github page) and paste here:

https://nbviewer.jupyter.org/