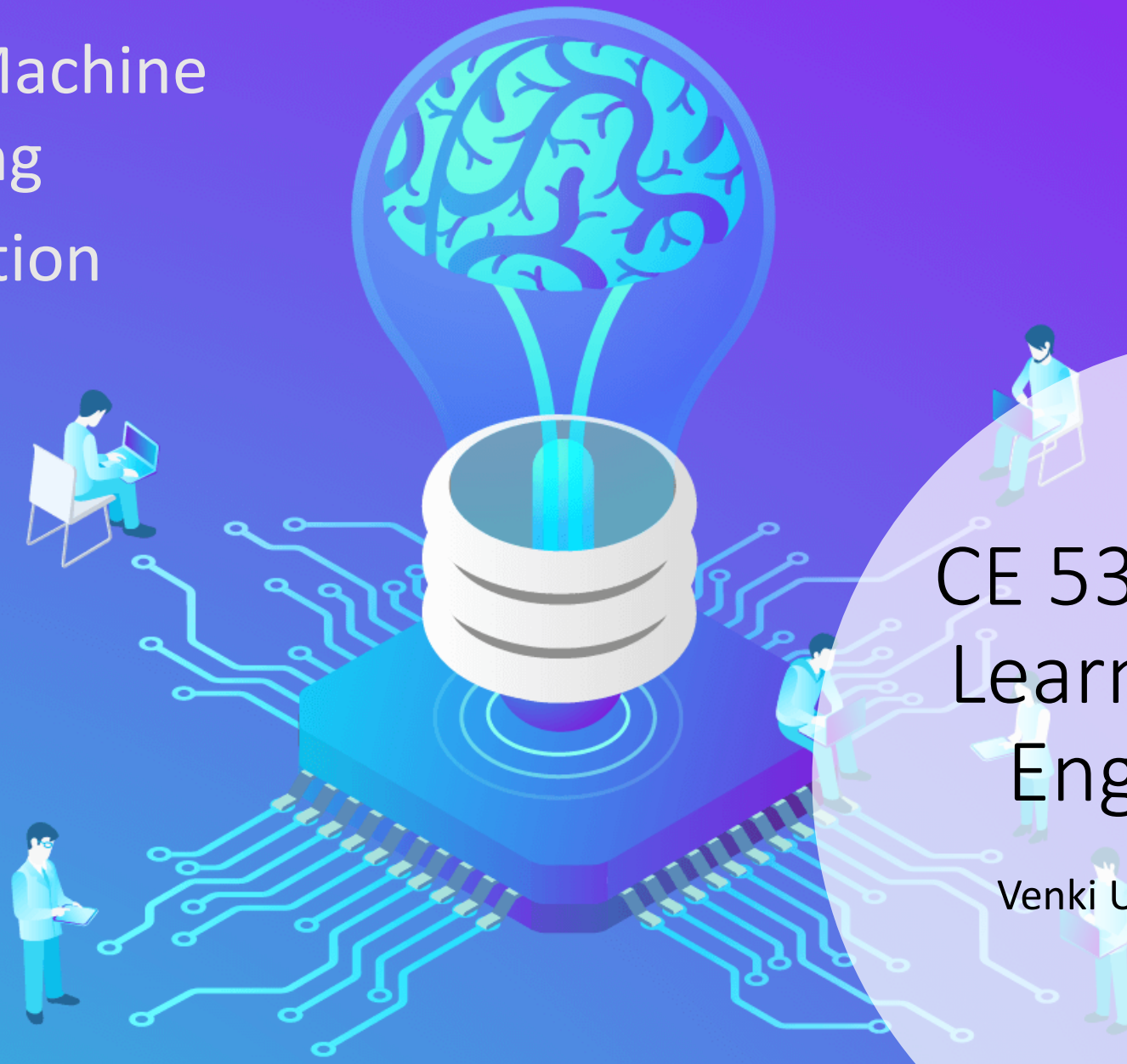
Python for Machine Learning

Visualization



CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap and Goals

- Installed Python and Anaconda Environments
- Introduction to Python
  - Setting working directory
  - Adding comment lines
    - Docstrings
- Introduction to Pandas
  - Reading a csv
  - Extracting columns (attributes)
  - Extracting rows
  - Obtaining summary measures

- Control Statements
  - If, if-elif-else, if-else
  - For loop
  - While loop
  - Use of Boolean operators
- Functions
  - Passing inputs
  - Lambda functions
  - Pass by object reference
- Numpy
  - Matrix Calculations
  - Vectorization

Goal of this module is to explore Plotting in Python

Plotting in Python

Plotting and Visualization is an important component of any Machine Learning project

Visualization must be carried out at all stages of the project

- Early stages to perform exploratory data analysis
- Visualize results for analysis
- Make high-quality visuals for presentation in reports and powerpoints

Integration of visualization within programming environments is therefore handy

Python - Visualization

- Native python is lean and does not come with significant visualization capabilities
- There are several packages that bring and enhance visualization capabilities in Python
  - Matplotlib is a very popular package for scientific visualization
  - Seaborn is another popular package
  - R's ggplot has also been ported to Python
  - Bokeh is a native python implementation of grammar of graphics (gg in ggplot)

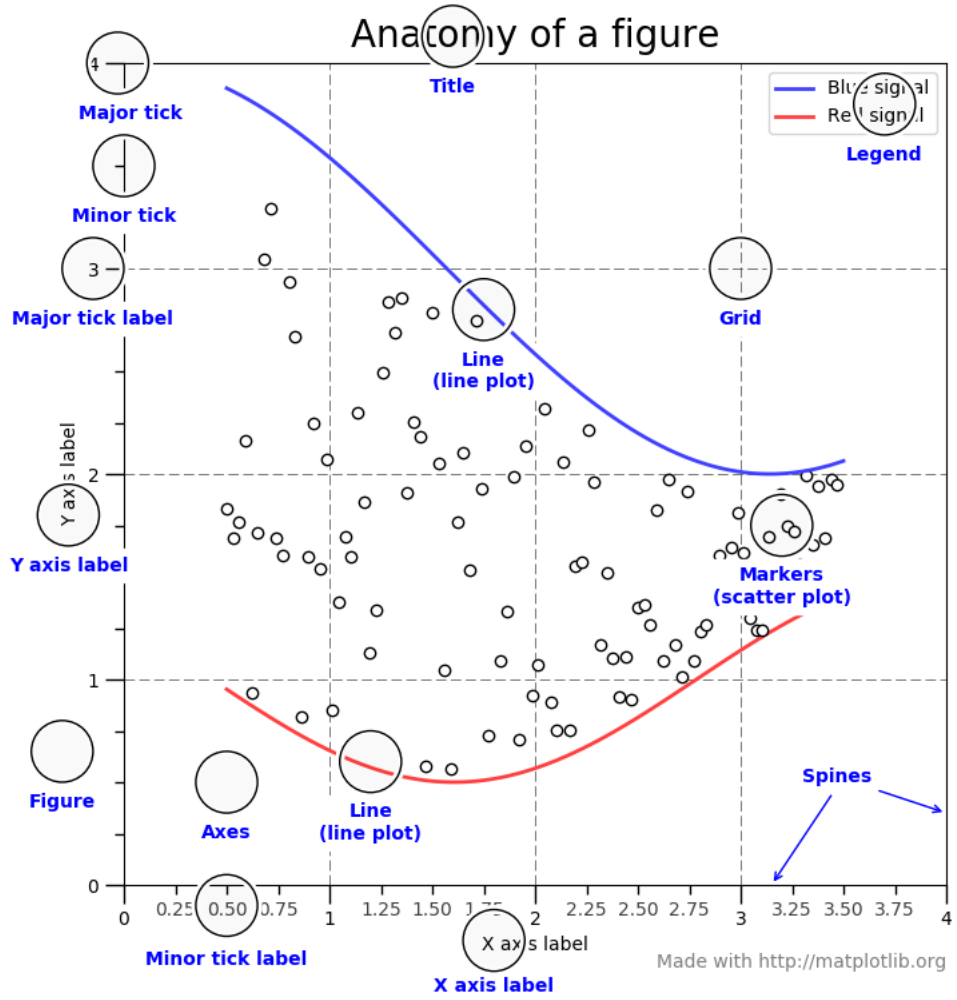
https://mode.com/blog/python-data-visualization-libraries

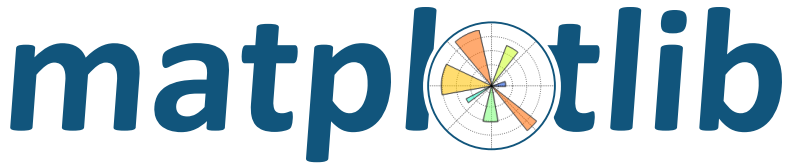
Visualization Philosophy

- Python packages follow similar visualization philosophy as R
  - Distinct from the visualization philosophy of EXCEL
- Python visualization libraries provide complete control to the user to render the plot to their specifications
  - Need to understand the elements of the plot and the necessary syntax
    - A steep learning curve (well worth the effort!!)
  - EXCEL creates a quick-and-dirty plot
    - Gives you very limited customization options
    - Short learning curve but not high quality

Elements of a Graph

- Even a basic figure has many components
  - Title
  - Tick marks
  - Axis labels
  - Markers
    - Style, size, color
  - Lines
    - Style, line width, color
  - Legend
  - Canvas
  - Frame
    - Grid, spines





MATPLOTLIB

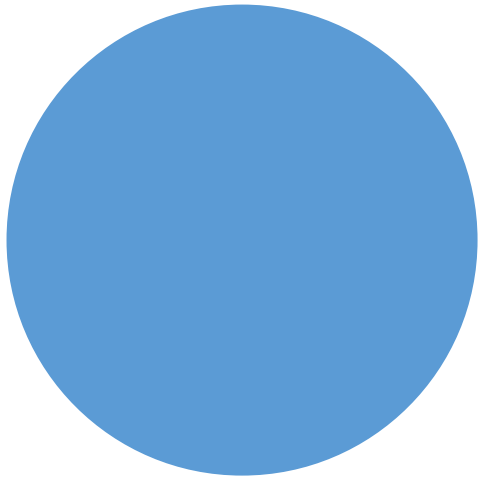
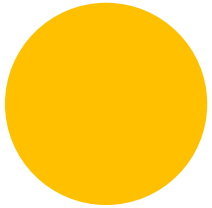
An Introduction

Matplotlib

- Matplotlib is a popular 2D plotting library for Python
- Allows you to make a variety of different graphs and charts
  - Scatterplots, bar plots, error charts, power spectra, etc.
- Plots can be made both in static (hardcopy) and interactive (web) modes
- Matplotlib uses MATLAB type interface
- Builds on numpy library
  - Uses numpy arrays for making plots
  - Sometimes best to convert your data to numpy arrays prior to plotting
    - Getting better with internal conversions
- The pyplot module of matplotlib library has many plotting functions that we shall use
  - You can simply import this module instead of the entire library

Matplotlib

- Matplotlib is a vast library with extensive functionalities
- My goal here is to get you started on some basic charting capabilities
- I will come back to some advanced plotting as and when necessary
  - Time-series data
- You can explore Matplotlib at your own leisure
- There is a wealth of information on the Matplotlib homepage
  - https://matplotlib.org/3.1.0/index.html

Pyplot Interface

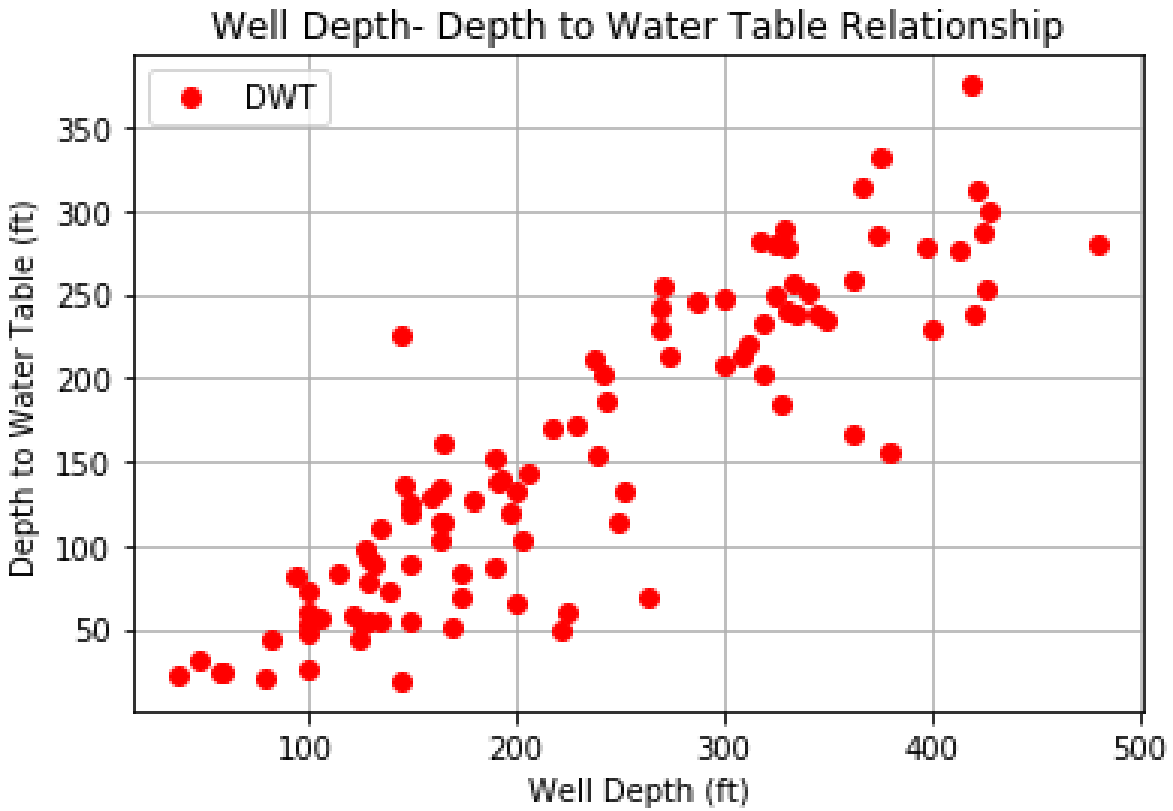
An Introduction

Plot

- Plot is a versatile command that can be used to make scatter and line plots between X and Y
- The following basic operations are necessary to make a scatter plot
  - Plot X and Y
  - Set the color and style of the points to be displayed
  - Set the limits of X and Y axes
  - Label X and Y axes
  - Add a title
  - Add a legend

Use the well depth and depth to water table data in OgallalafinalData.csv to make a scatter plot between Well Depth (X axis) and Depth to Water Table (Y-Axis)

In and X-Y Scatter Plot both X and Y are continuous variables



One can also use scatter function to make scatter plots

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.scatter.html

```
# Plotting scripts
# Venki Uddameri, 12/31/2019

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Read the OgallalaData.csv file
a = pd.read_csv('Ogallaladata.csv')
a.columns

# Create a scatter plot between Well Depth and Depth to Water-Table
DWT = a.DWT  # Extract DWT
WD = a.WellDepth # Extract well depth
plt.plot(WD,DWT,'ro') # Make plot (red color, circle marker)
plt.xlabel('Well Depth (ft)') # Add xlabel
plt.ylabel('Depth to Water Table (ft)') # Add ylabel
plt.grid() # Add grid
plt.title('Well Depth- Depth to Water Table Relationship') #title
plt.legend(['DWT'],loc='upper left') # Plot the legend
plt.show() # Show the graph on the console
```

Colors, Markers and Linestyles - Pyplot

For complete list of markers see
https://matplotlib.org/3.1.1/api/markers_api.html
#module-matplotlib.markers

- b: blue
- g: green
- r: red
- c: cyan
- m: magenta
- y: yellow
- k: black
- w: white

lineStyles =
{'-': '_draw_solid', '--': '_draw_dashed', '-.': '_draw_dash_dot', ':': '_draw_dotted', 'None': '_draw_nothing', ' ': '_draw_nothing', '': '_draw_nothing'}

marker	symbol	description
"."		point
","		pixel
"o"		circle
"v"		triangle_down
"^"		triangle_up
"<"		triangle_left
">"		triangle_right
"1"		tri_down
"2"		tri_up
"3"		tri_left
"4"		tri_right
"8"		octagon
"s"		square
"p"		pentagon
"P"		plus (filled)
"*"		star
"h"		hexagon1
"H"		hexagon2
"+"		plus

For additional information on colors see - https://matplotlib.org/2.0.2/api/colors_api.html

Making Multiple Scatter Plots – 1D stacking

- Make a vertically stacked plots for Average NO3 (NO3Av) and Depth to Water Table (DWT) against Well Depth (WD)
  - As they share the same X-axis only label the bottom X-axis
- The subplot method can be used for making staked plots
  - Also the preferred method for making single plots as per the matplotlib documentation

Code

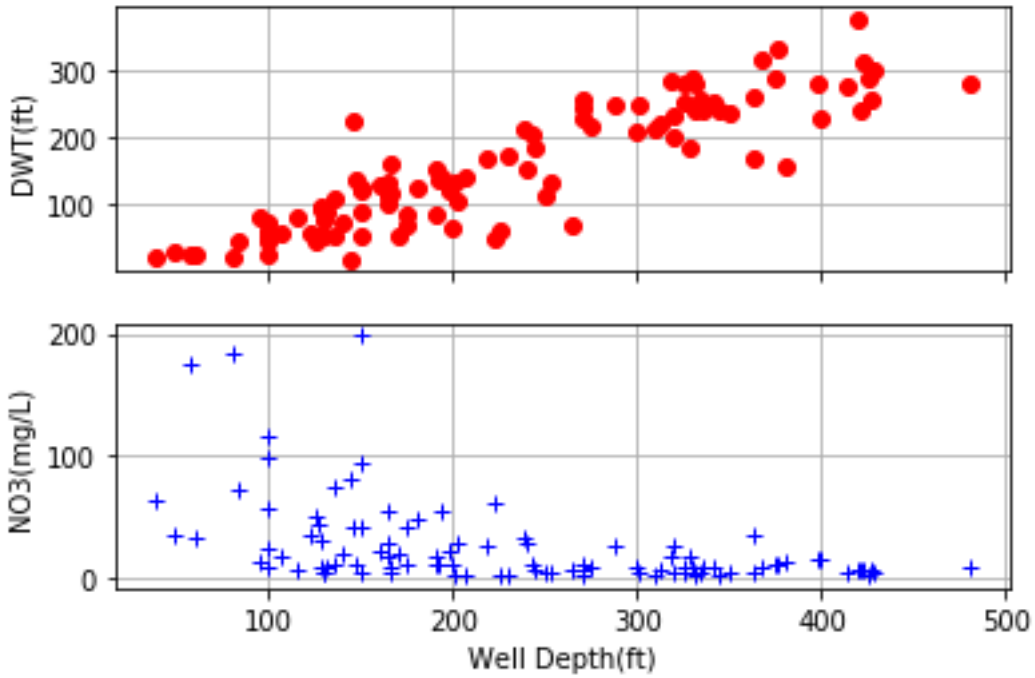
```
# Plotting scripts
# Venki Uddameri, 12/31/2019

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Read the OgallalaData.csv file
a = pd.read_csv('Ogallaladata.csv')
a.columns

# Create a scatter plot between Well Depth and Depth to Water-Table
DWT = a.DWT  # Extract DWT
WD = a.WellDepth # Extract well depth
NO3 = a.NO3Av # Average Nitrogen

# Create a 2 x 1 plot; Share same X-axis
fig,axs = plt.subplots(2,sharex=True) # Need two vertically stakcked subplots
axs[0].grid() # Add grid to plot 1
axs[1].grid() # Add grid to plot 2
axs[0].plot(WD,DWT,color='r',marker='o',linestyle='none') # Plot 1
axs[1].plot(WD,NO3,color='b',marker='+',linestyle='none') # Plot 2
plt.xlabel('Well Depth(ft)') # Add X-axis label (Same for both plots)
axs[0].set(ylabel='DWT(ft)') # Y-axis label for plot 1
axs[1].set(ylabel='NO3(mg/L)') # Y-axis label for plot 2
```



Multiple Plots

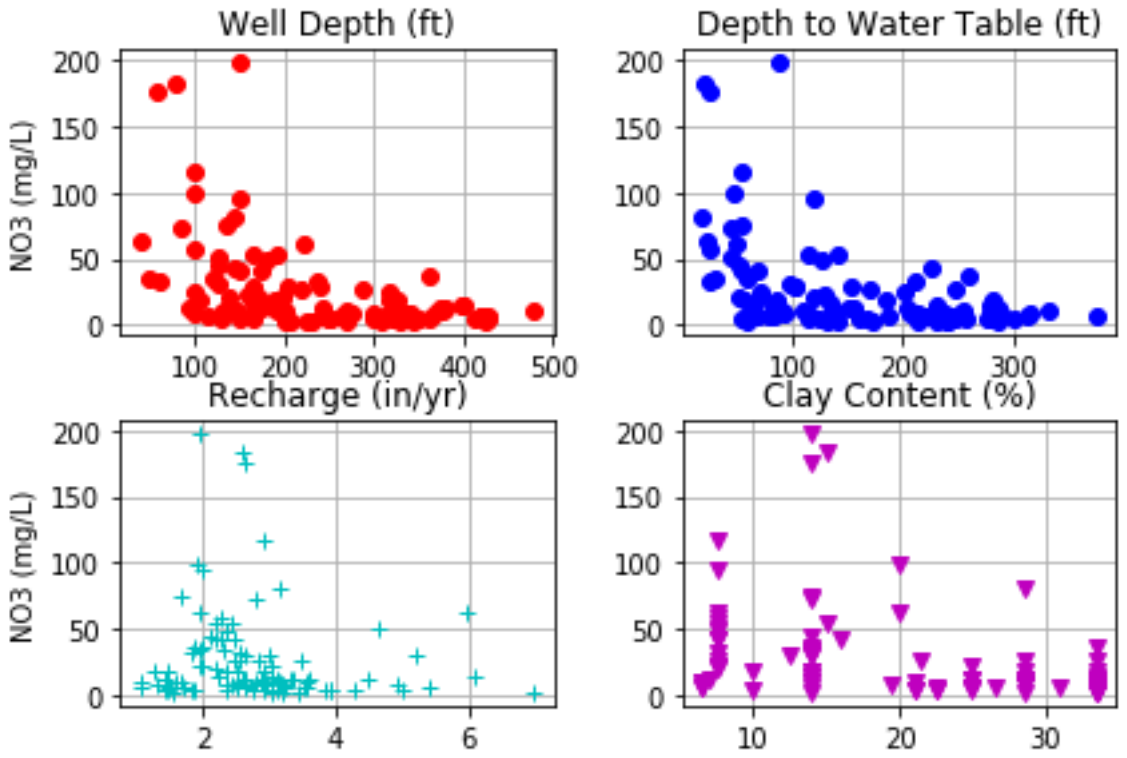
```
# Create a 2 x 2 plot
fig,axs = plt.subplots(2,2)
fig.tight_layout() # for good spacing between plots

axs[0,0].plot(WD,NO3,'ro')
axs[0,0].set(ylabel='NO3 (mg/L)')
axs[0,0].grid()
axs[0,0].set_title('Well Depth (ft)')

axs[0,1].plot(DWT,NO3,'bo')
axs[0,1].grid()
axs[0,1].set_title('Depth to Water Table (ft)')

axs[1,0].plot(a.Recharge,NO3,'c+')
axs[1,0].set(ylabel='NO3 (mg/L)')
axs[1,0].grid()
axs[1,0].set_title('Recharge (in/yr)')

axs[1,1].plot(a.Clay,NO3,'mv')
axs[1,1].grid()
axs[1,1].set_title('Clay Content (%)')
# Save the plot as a png (Change extension as required)
fig.savefig('4plot.png', dpi=450,papertype='letter')
```

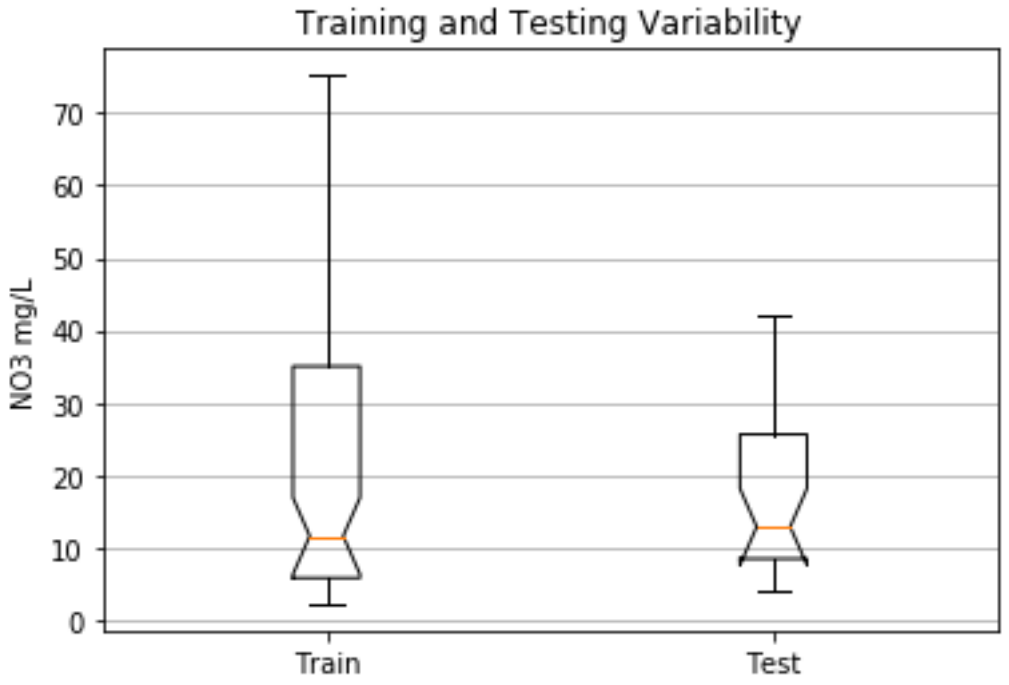


Use gridspec to get custom layouts to https://matplotlib.org/3.1.1/tutorials/intermediate/gridspec.html

Box plots

- Show the variability in NO3 training and testing data using boxplots

```
# Split training and testng data
NO3train = a.loc[a['Train']=='Training',['NO3Av']]
NO3test = a.loc[a['Train']=='Testing',['NO3Av']]
# Convert to a numpy array
NO3train = np.array(NO3train)
NO3test = np.array(NO3test)
# Make Boxplots
plt.boxplot([NO3train,NO3test],1,"") # Notched, No outliers
plt.xticks([1, 2], ['Train', 'Test'])
plt.ylabel('NO3 mg/L')
plt.grid(axis='y')
plt.title('Training and Testing Variability')
```

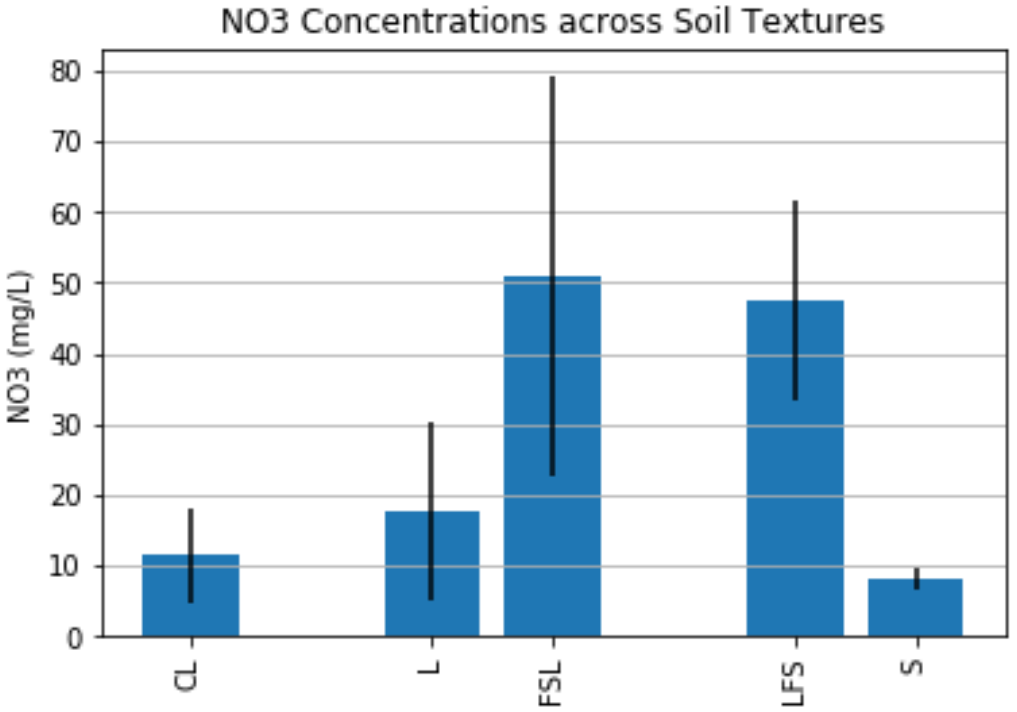


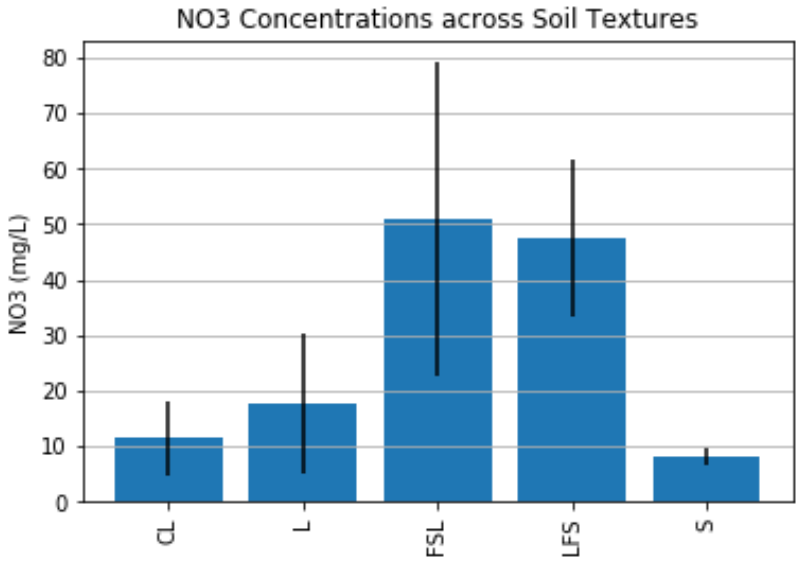
Boxplots provide a convenient way to compare variability across different groups

Bar Plots

- Bar Plots are used when comparing across categorical variables
- Compare NO3 concentrations across soil textures

```
# Make Bar Plot of texture versus Nitrate
b = pd.concat([a.Texture,a.NO3Av],axis=1) #create a NO3Texture DF
NO3avt = b.groupby(['Texture']).mean() # Find Group Mean
NO3std = b.groupby(['Texture']).std() # Find Group SD
X = list(NO3avt.index)
NO3avt = np.array(NO3avt.NO3Av) # Make a numpy array of means
NO3std = np.array(NO3std.NO3Av)*0.5 # Make a numpy array of 0.5 * sd
plt.bar(X,NO3avt,yerr=NO3std) # Make the barplot
tck = ['CL','L','FSL','LFS','S'] # tick labels
plt.xticks(X, tck,rotation='vertical') # label ticks
plt.grid(axis='y') # Add grid along Y-axis
plt.ylabel('NO3 (mg/L)') #Plot Y-axis label
plt.title('NO3 Concentrations across Soil Textures') # add title
```

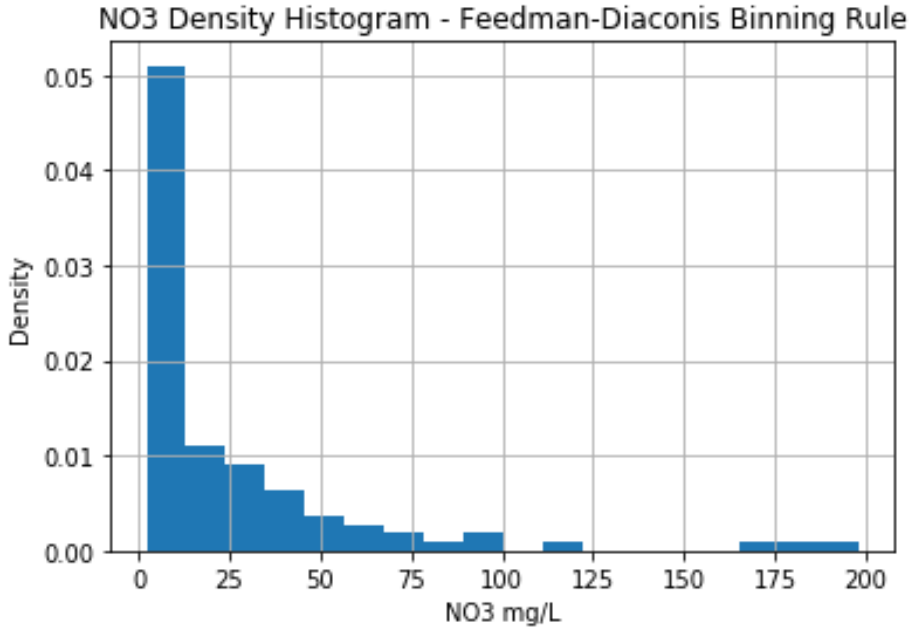




Note that soil texture codes are [3,5,6,8,9] Hence gaps are seen in the bar plot at [4.7]
You can recode the soil texture as [1,2,3,4,5] to remove the gaps

Histograms

- Histograms in Matplotlib pyplot
  - ‘pyplot.hist’ function
  - Can do density, count and cumulative histograms
- Based on histograms computed using numpy library
  - ‘numpy.histogram’
  - A variety of binning rules can be provided
    - Sturges, FD, Scott, etc.




```
#Plot Histogram
plt.hist(NO3,density=True,bins='fd') #Feedman-Diaconis Binning
plt.title('NO3 Density Histogram - Feedman-Diaconis Binning Rule')
plt.grid()
plt.ylabel('Density')
plt.xlabel('NO3 mg/L')
np.histogram(NO3,bins='fd',density=True) # this gives numbers
```

https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.histogram.html

Closing Remarks

- There are other packages in Python that provide graphing capabilities
  - Seaborn, scipy, statsmodels etc.
- Python's charting and graphing capabilities have come a long way in recent years
  - R continues to excel in this area
  - One approach would be to use R to make final plots
- Python visualization however shines for making online and interactive plots
  - Check out plotly library

You should know

- Basic visualization capabilities offered by Python
- How to make the follow plots
  - Scatter plot
  - Bar plot
  - Histogram
- How to layout multiple plots on a single sheet