Lab 05 – matplotlib I

Submission: `lab05.ipynb`, pushed to your Git repo on `master`.
Points: 10
Due: Friday, February 1, 9:59am

Objectives

- The basics of `matplotlib`

Introduction

Create a **`lab05.ipynb`** file. Create your header cell, then your first cell to set up your imports:

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

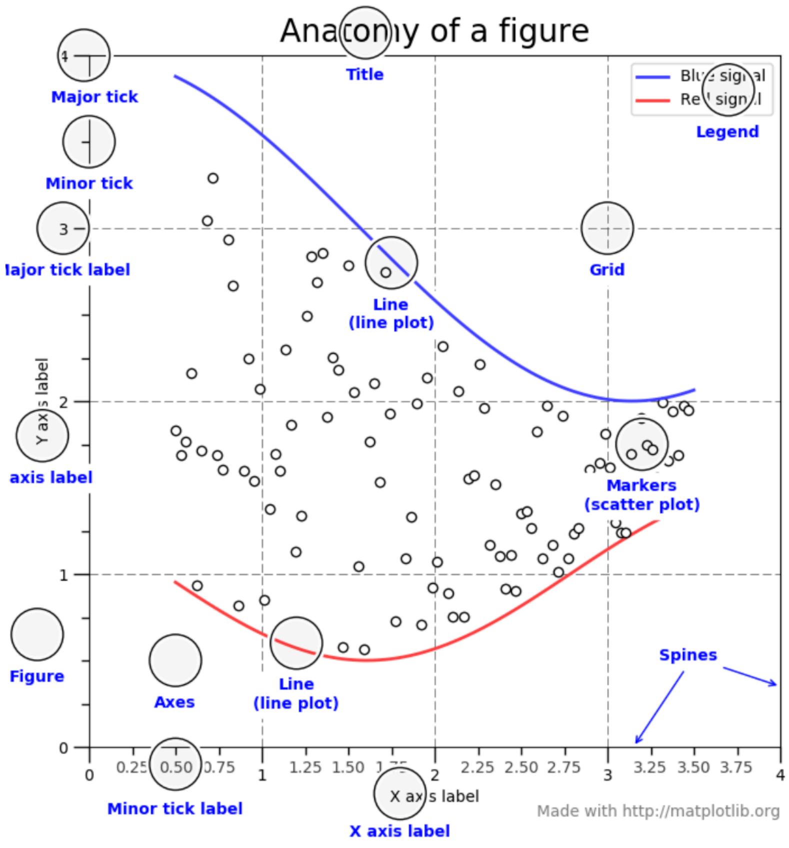
First off, before you begin these exercises, if you are not familiar with matplotlib, I strongly recommend that you work through the short, concise, but highly effective beginner's tutorial on matplotlib:
https://matplotlib.org/tutorials/introductory/pyplot.html

Then, page through some of these tutorials and example code, and remember this page. It has a wealth of example code you'll be needing for the rest of the semester: https://matplotlib.org/tutorials/index.html

You might as well bookmark this API reference, because you are going to be referring to this API perhaps more than any other page this semester. (OK, probably not, but it will feel like it at times.)
https://matplotlib.org/api/index.html

Did you bookmark this direct reference to the pandas API page for the `DataFrame` class yet?
https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

And, this diagram is your sanity as you begin to wrap your head around this visualization framework. The names that are used to label the important entities on the plot below matches the names used in the API:

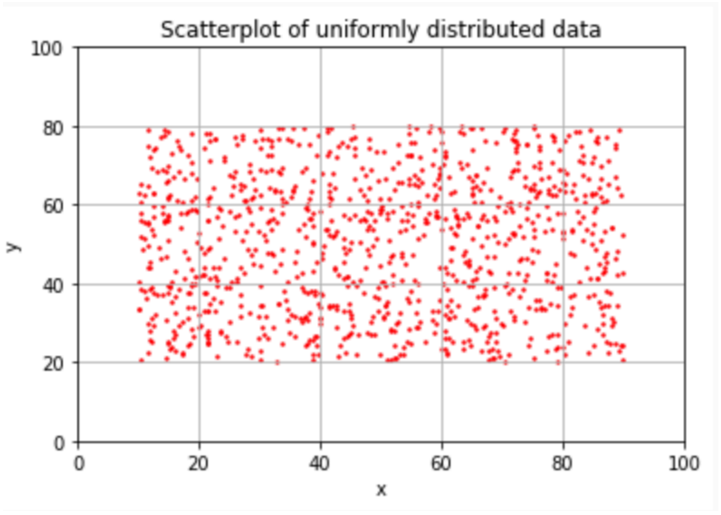


Part I - matplotlib

1) [P] Create a dataframe named `df_uniform` that contains 1000 observations. It should have two variables, named `x` and `y`. For each observation, `x` should be generated from a uniform distribution between 10 and 90, and y should be generated from a uniform between 20 and 80.

2) [P] Generate a scatterplot of the data. Your plot must:
   a. Have a title
   b. Label both axes with `"x"` and `"y"` respectively
   c. Change the x and y axis to display between 0 and 100
   d. Change the default point size
   e. Change the default color of the point
   f. Display a grid

Here is one possible example:

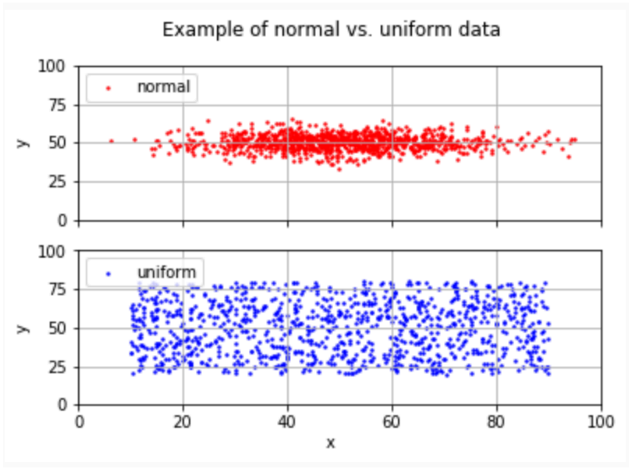


3) [P] Generate a data frame called `df_normal` with 1000 observations, two variables names x and y again. This time, x should be generated from a normal distribution with mean 50 and standard deviation 15, and y with mean 50 and standard deviation 5.

4) [P] Repeat your plot above with `df_normal`, but use a different color point, and title your plot accordingly.

5) [P] Generate a single figure that contains two axes that are adjacent to each other. You should have:
   a. at least one shared axis
   b. appropriate axis labels
   c. make the range of the axis on both plots the same
   d. display a legend on each to be sure both are labeled correctly as "normal" or "uniform"
   e. One title at the top

   Here is one example:



6) [P] Display both `df_uniform` and `df_normal` on one shared plot, with an appropriate legend

7) [M] What is a **histogram**? In your answer, please clearly indicate what it is, why one would use it during their EDA phase, and whether it's good for one variable or to show relationships between multiple variables.

8) [M] What is a **boxplot**? In your answer, please clearly indicate what it is, why one would use it during their EDA phase, and whether it's good for one variable or to show relationships between multiple variables.

9) [M] What is a **density** plot? In your answer, please clearly indicate what it is, why one would use it during their EDA phase, and whether it's good for one variable or to show relationships between multiple variables.

10) [P] The pandas `DataFrame` class has a useful interface to matplotlib that will help you generate some quick plots as you explore your data. To get you started, generate a **histogram** of both the x and y variables for `df_uniform`. Use 30 bins, and set the range of both variables to be 0 – 100. Repeat this exercise on `df_normal`.

11) [P] Repeat the previous exercise to generate a **boxplot** on both x and y variables of both `df_uniform` and `df_normal`.

12) [M] What is a quantile? What is a quartile? What is an Inter-quartile range (IQR)? Interpret the boxplot results in these terms.

13) [M, P] Read about the `quantile()` method for data frames, and use it to numerically show the 25th, median, and 75th percentiles, and compute the IQR (Inter-quartile range) for both variables, on both data frames. Compare and contrast.

14) [M, P] What is the standard definition of an outlier, in terms of IQR? Use the IQR to determine what the outliers are for each variable in each dataset, if any.

15) [P] Generate a density plot for both x and y variables of both `df_uniform` and `df_normal`.

16) [M] Interepret the density plot results

17) [P] Go back to the `describe()` method you learn about in previous labs. Show the results of `describe()` for both data frames.

18) [M]] What is a quantile-quantile plot?

19) [P, M] Load the `scipy.stats` package, and read about the `probplot` function. This can generate a QQ plot for you. Use it to generate a plot for `df_uniform.x`, and `df_normal.x`. assume your distribution is normal for both (even though we know it is not!) Compare and contrast your resulting plot. Does the output suggest that one is indeed normally distributed, and the other is not?

Part II - seaborn

For this next part, you are going to install an additional package into your csci349 conda environment. Install **`seaborn`**, which is a data visualization package built on top of matplotlib. (You can easily install this package from Anaconda Navigator, or using `conda` from the command line.) Matplotlib is a solid graphics engine that can handle a LOT of different techniques. Seaborn takes matplotlib to the next level of modernization, makes many plotting tasks easier, and generally, it adds some seriously nice attractiveness.

Before you begin, please take the time to read through https://seaborn.pydata.org/introduction.html. Then, just so you have a sense of what some of its capabilities are, browse through the various tutorials on https://seaborn.pydata.org/tutorial.html . There are a LOT of different tutorials here. Seaborn focuses on helping you plot statistical relationships in your data, which is precisely what we're after when we are doing EDA with our data!

20) [P] Add the following import statement:

```
import seaborn as sns
```

21) [P] Show a single scatterplot of `df_normal` using `sns`. Change the default color and point type that is used in the plot.

For the remainder of these exercises, you are required to use seaborn, but select at least two aspects of your plot to make them unique. It could be the color of the point, size, background, grid, etc. etc. There are many choices. Use these exercises to learn about this wonderful visualization framework, and to tap into the artist in you!

22) [P] Show a scatterplot of both `df_uniform` and `df_normal` side by side on the same figure

23) [P] Show the distribution of only the x variable for both `df_uniform` and `df_normal`, with a density curve and a rugplot at the bottom. (Look at `sns.distplot`)

24) [P] Use `sns.jointplot` to show the bivariate distribution of `x` and `y` for `df_uniform` and `df_normal`

25) [P] Show a hexbin plot using `sns.jointplot` for `df_normal`

26) [P] Show a kernel density estimation (kde) using `sns.jointplot` for `df_normal`

Part III – Some basic data preprocessing

27) [P] Create an additional variable in `df_uniform` called `x_fac1` that represents a factor with 3 levels, "X1", "X2", and "X3". You should discretize according to equal width bins over the distribution of x. (Divide the range of x into three.)

28) [P] Create an additional variable in `df_uniform` called `x_fac2` that represents a factor with 3 levels, "X1", "X2", and "X3". This time, you should discretize using equal depth bins over the distribution of x. Select your division criteria such that there are an equal number of data in each bin. Verify that the distribution of your data each has the same number of data (within 1).

29) [P] Create a side by side scatter plot showing the distribution of `df_uniform`, using `x_fac1` as the color for one plot, and `x_fac2` as the color for your other plot.

Deliverables

Commit and push `lab05.ipynb`. Be sure you have every cell run, and output generated. Verify that your file is pushed properly on Gitlab.