# In-class assignments for Section 1: Data Exploration and Presentation

Subhasis Ray*

*<2022-09-29 Thu>*

- Save your work in a file named `W05P1_classwork.py` and submit it to codePost.
- The first deadline for submitting your work on codePost is 1:30 PM. Do this to get attendance.
- The submission will open again in the afternoon, and you have until midnight to submit an updated version. Also submit the figures (saved files) on LMS.
- Do not worry about making your code look clean. Leave any comments/docstring you wrote intact when submitting.
- Look at matplotlib tutorial here: `https://matplotlib.org/stable/tutorials/introductory/pyplot.html`
- You can see a gallery of plots them here: `https://matplotlib.org/stable/gallery/index.html`, they have links to the Python code for reproducing the plots, too.

## 1 Startup

In this assignment we shall explore a sample dataset on salaries by type of college in the USA using pandas and matplotlib.

Import numpy, pandas, and matplotlib.

*subhasis.ray@plaksha.edu.in

```
#%% imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Load the data as a pandas dataframe called 'df'.

```
#%% load data
df = pd.read_csv('salaries-by-college-type.csv')
```

This data has the salary columns with the $ symbol, and separated at thousand's place by comma. So this is a string column. In order to do numerical calculations, you have to convert them to numbers. Moreover, since the original column names are long and descriptive, they are a pain to type. Moreover, as these names contain space and other illegal characters for Python identifiers, you cannot access the columns as attributes (you have to write `df['col']` instead of `df.col`, which takes less typing). So we shall assign the numeric values to new column names.

Now you know how to add a new column to a DataFrame.

```
#%% Convert strings with $ and , to numbers
df['starting'] = df['Starting Median
↪  Salary'].str.replace('[$,]', '', regex=True).astype(float)
df['mid'] = df['Mid-Career Median Salary'].str.replace('[$,]',
↪  '', regex=True).astype(float)
df['mid10'] = df['Mid-Career 10th Percentile
↪  Salary'].str.replace('[$,]', '', regex=True).astype(float)
df['mid25'] = df['Mid-Career 25th Percentile
↪  Salary'].str.replace('[$,]', '', regex=True).astype(float)
df['mid75'] = df['Mid-Career 75th Percentile
↪  Salary'].str.replace('[$,]', '', regex=True).astype(float)
df['mid90'] = df['Mid-Career 90th Percentile
↪  Salary'].str.replace('[$,]', '', regex=True).astype(float)
```

We shall also rename the 'School Name' and 'School Type' columns:

```
#%% Rename columns to something shorter
df.rename(columns={'School Name': 'sname', 'School Type':
↪  'stype'}, inplace=True)
```

Just to reduce clutter, and to see how to drop columns, we shall remove the original salary columns in string format:

```
#%% Rename columns to something shorter
df.drop(columns=['Starting Median Salary',
                 'Mid-Career Median Salary',
                 'Mid-Career 10th Percentile Salary',
                 'Mid-Career 25th Percentile Salary',
                 'Mid-Career 75th Percentile Salary',
                 'Mid-Career 90th Percentile Salary'],
                 ↪  inplace=True)
# See what columns we got now
print(df.columns)
# Print the head of the dataframe
print(df.head())
```

Did you notice the NaN entries? The data was `N/A` in those. These are actually numpy `nan` values.

## 2 Line plot: 5 points

One of the most basic plots is a line plot. Do this with some random numbers.

```
#%% Create a random number sequence
rng = np.random.default_rng(0)
y = rng.random(10)
```

Also, to have different things plotted in different windows, you have to create figures. A figure is a single image, it can contain multiple axes. A versatile way to create a figure with axes is the following:

```
#%% Create a random number sequence
fig, ax = plt.subplots()
```

1. Now plot the `y` values by simply calling `ax.plot()` function with `y` as the first argument. Also specify the label of the plot using the `label` keyword argument.
2. Add a legend using the `ax.legend()` function.
3. Add a title with `fig.suptitle()`.
4. Set the range of the X axis from -1 to 12 using `ax.set_xlim()`.
5. Set the tick positions on Y axis to 0.1, 0.3, 0.5, and 0.7 using `ax.set_yticks()`.
6. Save your figure as `line_plot.png` using `fig.savefig()`

## 3 Pie chart (5 points) and bar plot (5 points) with titles and labels (5 points)

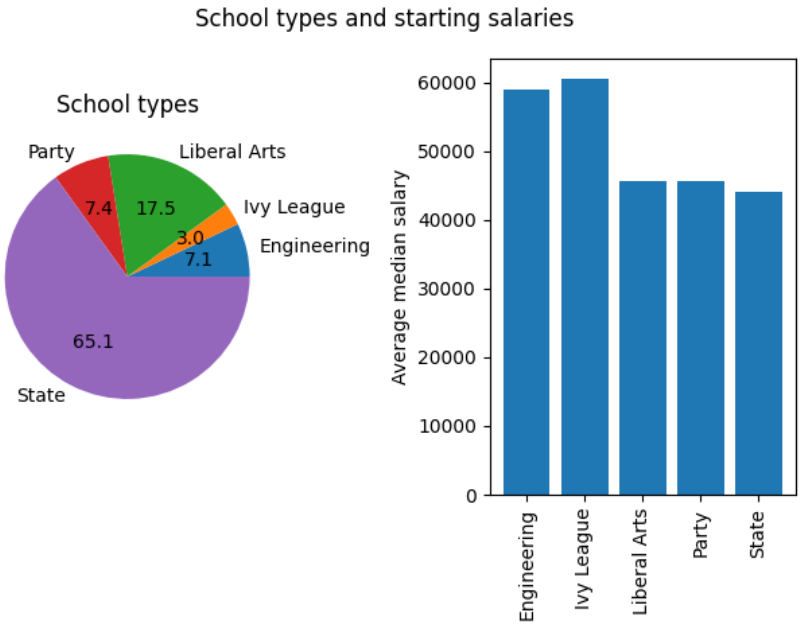
Now we shall plot a pie chart and a bar chart side by side. For this you need two axes. The way to get that is to specify the number of rows/columns with the `nrows` and/or `ncols` arguments to `plt.subplots` function. Note that now your `ax` is a 1D array, so you have to get the axes using index, like `ax[0]`, `ax[1]`, etc.

Plot a pie chart of the proportion of different college types in our data. You can use `df.groupby().agg()` to get a count of colleges by 'School Type' (renamed to `stype` above). This returns a dataframe where the `index` is `stype`. This dataframe has the same columns except the grouping column `stype`, but the values are the counts for that school type.

Pass the counts in any of these columns as the first argument to `axes.pie()` and the labels (`counts.index`) as the keyword argument `labels`. Also pass `autopct='%1.1f'` to display the percentages in the pie chart. Set an axis title using `axes.set_title()`.

Now in the second axes create a bar chart showing the average starting salaries (`starting`) for each school type. Again you can use `groupby()` and `agg('mean')` to do this. Use `axes.bar()` function for this. Set a label for the Y-axis, say `"Average median salary"` using `axes.set_ylabel()`.

If the labels on the X-axis become cluttered, you can rotate them with `axes.tick_params(axis='x', rotation=90)`. If after rotation, parts of your labels get cropped, call `fig.tight_layout()` to automatically adjust the figure so everything fits. Save the figure as `pie_bar_plot.png`.

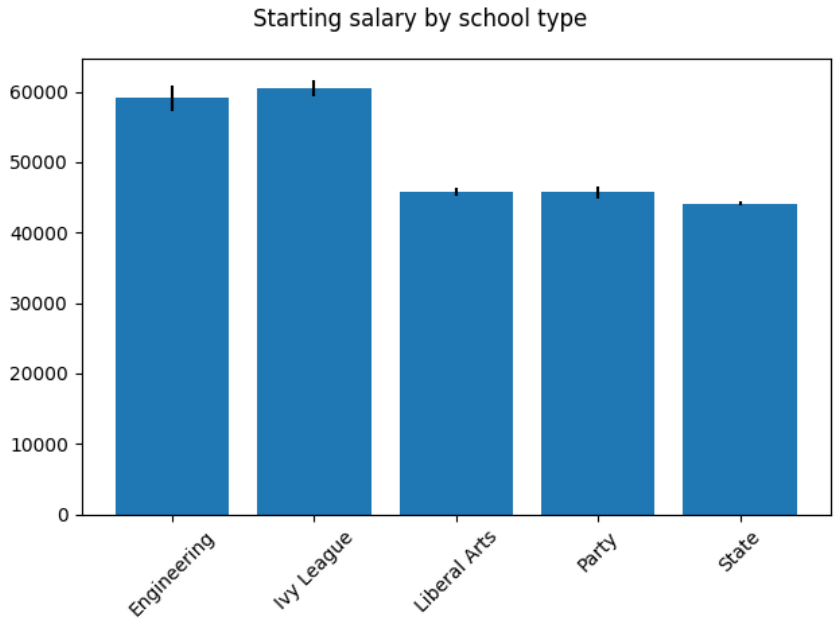



## 4 Bar chart with error bars: 5 points

Now create a new figure with a single axis. Plot the same bar chart as above. However, this time you have to put error bars on each bar. This error bar should show the SEM (standard error of the mean), which you can obtain with `df.groupby().agg('sem')`.

You can plot the error bars with bar plot by specifying the `yerr` argument.

Save the figure as `error_bar_plot.png`.

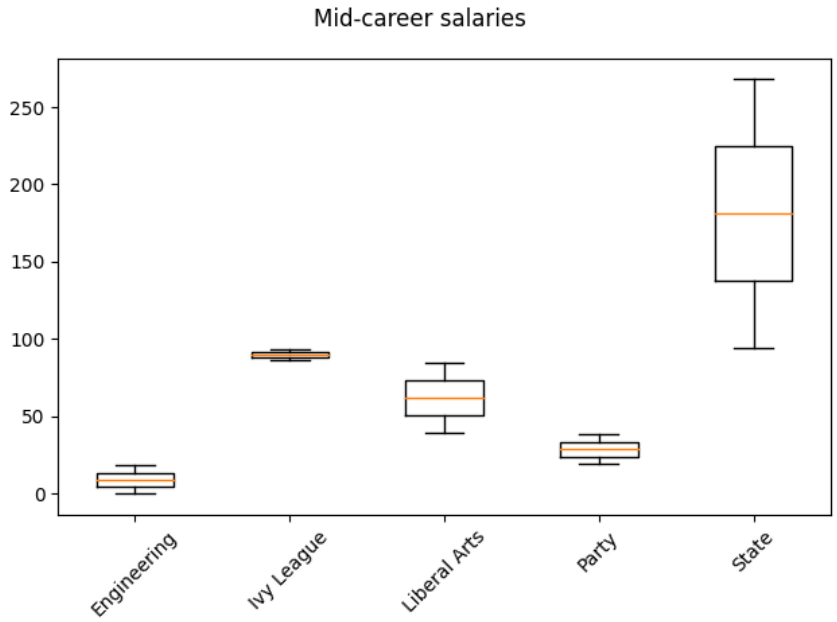



## 5 Box plot the mid career salaries: 5 points

Create another plot showing mid-career salaries using `axes.boxplot()` with the data grouped by schooltype. Pass the `stype` as `label` keyword argument so that each box has the school type on X axis. After a group by, you can get a dict-like mapping from the key (`stype`) for each group to the values in that group via `groupby().groups`. For example,

```
#%% roups
mid = df.groupby('stype')['mid']
values = mid.groups.values()
stypes = mid.groups.keys()
```

Save the figure as `box_plot.png`.




## 6 Submit the figures on the LMS and your code on codePost.