Describing Data: University Rankings

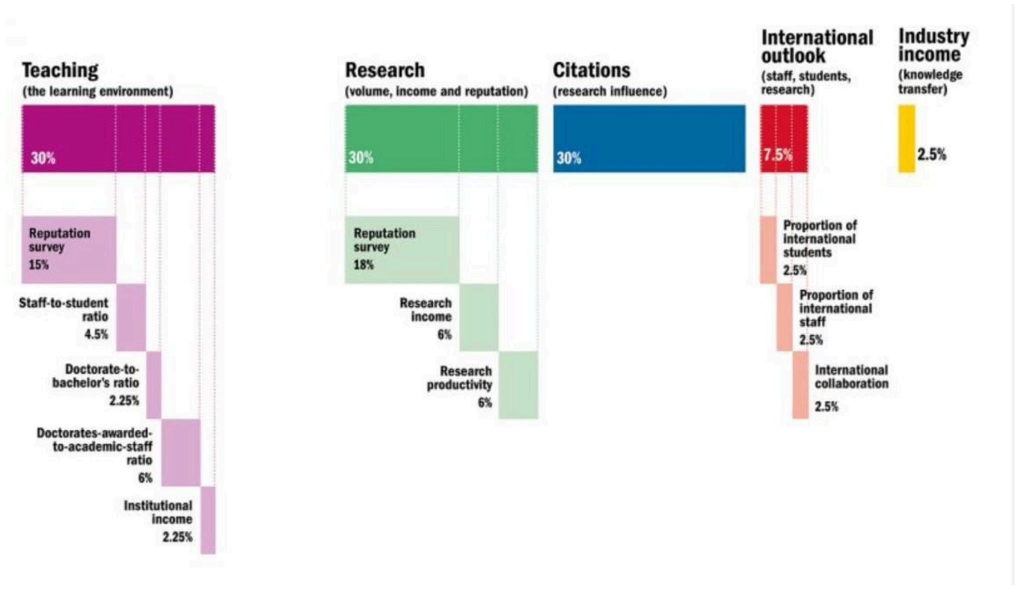
In this assignment, you will explore the Times Higher Education university rankings (https://www.timeshighereducation.com/world-university-rankings/2016/world-ranking/detailed). We are using rankings from 2016 as the data has been made available on Kaggle in an easy-to-use format (https://www.kaggle.com/datasets/mylesoneill/world-university-rankings). You have been given an adapted version of this data to use. There are three parts to the assignment. Part 1 and Part 2 must be submitted according to their specifications to earn a B (3.2) on the assignment. You may choose to submit Part 3 to earn an A on the assignment – be aware that you can only earn an A by completing the B-level components first (i.e., you shouldn't do this part unless your Part 1 and Part 2 are complete). Each part provides its own set of instructions. The general specifications for this assignment, which are required to achieve a passing grade, are:

- Answers to questions from all parts submitted as a single .pdf, images submitted as .png files exported from code (no screenshots) with names as indicated, code for all sections submitted as a single .py or .ipynb file.
- .pdf questions file and code file labeled clearly throughout as to which part corresponds to which question/task.
- Files submitted are readable in Blackboard (except .ipynb files)– this means you must check your submission after turning it in.
- Free of spelling, capitalization, and large-scale grammatical errors.
- For a group submission: only one submission in Blackboard by one group member with other members listed in the submission comments (if you don't know how to do this, please ask).
- Work is your own/your group's own, with sources cited – class notes and official Python documentation do not need to be cited; any unauthorized collaboration/sources or use of allowed sources without citation invalidates the submission.

The instructions that follow are lengthy. They tell you exactly what to do, so read them carefully and consult them as you complete your work. I would suggest using the specifications as a checklist – go over them several times to make sure you have met them in full.

Part 1: Zooming in on the Top 200

The Times Higher Education (THE) University Rankings evaluates universities around the world based on Teaching, Research, Citations, International Outlook, and Industry Income:



Teaching
(the learning environment)
30%
Reputation survey 15%
Staff-to-student ratio 4.5%
Doctorate-to-bachelor's ratio 2.25%
Doctorates-awarded-to-academic-staff ratio 6%
Institutional income 2.25%
Research
(volume, income and reputation)
30%
Reputation survey 18%
Research income 6%
Research productivity 6%
Citations
(research influence)
30%
International outlook
(staff, students, research)
7.5%
Proportion of international students 2.5%
Proportion of international staff 2.5%
International collaboration 2.5%
Industry income
(knowledge transfer)
2.5%

The data that THE provides includes two metrics in the ranking scale, the proportion of international students and staff-to-student ratio, and two metrics not included in the rankings, the number of students and the female-to-male ratio. We will first look at the top 200-ranked schools in 2016 and how their overall score relates to these four parameters. We will also highlight the top 10 schools in our visualizations to look for patterns in these metrics. The file `College Data.csv` contains THE rankings from 2016 obtained from Kaggle. Use this data file; do not download the original data.

To prepare for making the visualization, it is suggested that you create two new data frames from the original: 1) a data frame containing the top 200 colleges only (hint: use `iloc` from 0 to 200); 2) a data frame containing the top 10 colleges only (hint: use `iloc` from 0 to 10). It is okay that these data sets overlap. You can also make the visualization without doing this, but it may be slightly more difficult.

Task 1

Create a set of scatterplots as a figure with four subplots in a 2x2 grid. Each scatterplot will have on the x-axis one of the following variables:

`num_students` (count of the total number of students)

`student_staff_ratio`

`international_students` (percent of international students)

`female` (percent of female students)

Add a vertical line to represent the median of the x variable on each plot. Every scatterplot should have the total score (`total_score`) on the y-axis. Further details and instructions are provided below.

A visualization that receives full credit will:

- Be a figure consisting of 4 subplots that contain scatterplots organized in a 2x2 grid as described above – you MUST create subplots; use the `gridspec` keywords `hspace` and `wspace` to create adequate space between the plots for labeling.
- Contain scatterplots that do not use the defaults but rather show enhancements and customization based on material learned in class.
- Plot the top 200 points using a marker of a suitable size and color so that the top 10 can be plotted on top of them; this means you will call `ax.scatter` twice for each plot: first plot the top 200, then plot the top 10 so that we see 200 markers with the top 10 called out in particular – do not remove the top 10 from the top 200 (hint: I did this by plotting the top 200 as circles, and then indicating the top 10 with stars that fit in the circles); ASK IF YOU DO NOT UNDERSTAND.
- Have colors for the plot based on the principles discussed in class and in the reading (hint: remember what Muth says about grey); you can see a complete list of named colors in matplotlib here: https://matplotlib.org/3.1.0/gallery/color/named_colors.html.
- Label all axes in title case, with meaningful names, and of appropriate font size as per our discussions in class and the readings; be sure to include additional labeling and remove default labeling as needed.
- Increase the font size of the tick labels for legibility, but keep them proportional to the axis labels.
- Include a vertical line on each plot for the median of the variable plotted on the x-axis (hint: use `axvline`); make sure to adjust the line color, weight, and style so that the line is visible but not distracting.
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `scatter.png`; save your plot to the file using `plt.savefig('name.png', bbox_inches = 'tight', facecolor = 'white');` check that your files saved correctly.
- Adhere to guidelines in class for readability (e.g., removing non-data ink), truthfulness (no data hiding or distortion), and effectiveness..
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses matplotlib for plotting; ideally, your code should include a loop, but if it does not, it should be thoroughly commented to indicate what is what.

Task 2

Answer the following questions in a few sentences (3-5 is fine. Longer is ok as well) as described below:

A. Calculate correlation coefficients for each of the x variables versus the total score. How do the correlation coefficients compare to what you see in the scatterplots? Are there any cases where the correlation coefficient could be misleading? Why or why not?
B. Based on your visualization, what do you observe about the top 10 schools for each of the four variables? Are they randomly scattered, or is there a discernible pattern? How do they relate to the rest of the top 200? [N.B.: you should not be commenting that they are all at the top of the plot….that is 100% expected…why?]

Part 2: Comparing the highest-ranked to the lowest-ranked

For this part of our exploration, you will need the top 200 data set you already created, and you should also create one that contains the last 200 (lowest-ranked) – again, you can use `.iloc` to create this. Join these two together using `pd.concat` (the code given assumes your data frames have the names `top_200` and `last_200`):

```
first_last = pd.concat([top_200,last_200]).reset_index()
```

Then, create a new column that marks whether the observation was in the top or last 200:

```
first_last['Ranks'] = ['top']*200 + ['last']*200
```

Task 1

Create a parallel coordinates plot that compares the top 200 and the last 200 based on the variables:

```
teaching        research        citations
income          female          international
```

Further instructions are provided below. A visualization that receives full credit will:

- Be a parallel coordinates plot made as described in class, using the variables specified above, with top vs last as the categorical determinant.
- Order the numeric variables so that it is easy for the eye to compare and highlight differences between the categories.
- Use colors and opacity that provide sufficient contrast and visibility.
- Have a custom legend created, as shown in class, that maximizes legibility.
- Label all axes in title case, with meaningful names, and of appropriate font size as per our discussions in class and the readings; be sure to include additional labeling and remove default labeling as needed.
- Increase the font size of the tick labels for legibility, but keep them proportional to the axis labels.
- Adjust gridlines to decrease distraction (as discussed in class).
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `parallel.png`; save your plot to the file using `plt.savefig('name.png', bbox_inches = 'tight', facecolor = 'white');` check that your files saved correctly.
- Adhere to guidelines in class for readability (e.g., removing non-data ink), truthfulness (no data hiding or distortion), and effectiveness.
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses pandas parallel_coordinates for plotting.

Task 2

Answer the following questions in a few sentences (2-5 is fine. Longer is ok as well) as described below:

A) We didn't standardize the data for the parallel coordinates plot. This is fine. Why?
B) Which variables show the largest difference between the top and last 200? Describe what you see in the plot briefly. Your answer should be based on your visualization, not any outside information.
C) What you saw in B) was largely expected…why is this? (Hint: use the graphic that shows how scores were calculated)

Part 3: This section is only required if you are seeking to earn an A on the assignment – you should only attempt this if you have completed the mandatory section.

Task 1

For this part of the assignment (both tasks), we will look at ALL of the data, not just a subset. Make sure you are using the FULL data to complete this. Since there are so many observations and they have similar values, it will be hard to visualize anything using a regular scatterplot. Instead, we will make a set of hexagonal binning plots (known as `hexbin`) that show the same information as in Part 1 of the assignment, but for the FULL data set. The official documentation is here: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hexbin.html. You have also been provided with a tutorial in the notebook `Hexagonal Binning Code.ipynb` – the expectation is that you will use the tutorial to create this visualization. You should be able to use your code from Part 1 and make a handful of changes (<10 changes if you used a loop, more if not).

A visualization that receives full credit will:

- Be a figure consisting of 4 subplots that contain hexbin plots organized in a 2x2 grid. As for Part 1, you MUST create subplots; use the gridspec keywords hspace and wspace to create adequate space between the plots for labeling.
- Contain plots that do not use the defaults but rather show enhancements and customization based on material learned in class and from the tutorial.
- Use a colormap that enhances contrast and is appropriate for this data type.
- Transform data as needed to avoid distortion (hint: are there any very large or very small outliers? If so, you might need to transform the data).
- Label all axes in title case, with meaningful names, and of appropriate font size as per our discussions in class and the readings.
- Increase the font size of the tick labels for legibility, but keep them proportional to the axis labels.
- Ensure that your figure is large enough that all components are visible and identifiable.
- Save the figure in .png format as `hex.png`; check that your file is saved correctly.
- Adhere to guidelines in class for readability (e.g., removing non-data ink), truthfulness (no data hiding or distortion), and effectiveness.
- Provide Python code used to generate the plot; code will run without errors to re-generate the plot; code uses matplotlib for plotting; ideally, your code should include a loop, but if it does not, it should be thoroughly commented to indicate what is what

Task 2

Based on your plot, answer the following question in a few sentences (3-5 is fine):

For each of the plots, where is most of the data concentrated in the binning plot? What does this mean in terms of the values of the variables? What trends (if any) do you see in the data? Which variables have no trends?