Problem Set 1

Advanced Methods in Applied Statistics 2

Kimi Cardoso Kreilgaard (TWN176)

February 16, 2022

Disclaimer: I worked with Linea Stausbøll Hedemark during the exercises, so some of our results and some of the code may bear similarities.

1 Exercise 1

1.1 Extracting the Data

To extract the data we use `pandas.read_html`. To make sure the file is unchanging I download the website as an html file for both 2009 and 2014. These files are submitted along with the code, and should be placed in the same directory. There are to thing to be aware of when reading in the files:

- The html file contains a lot of unnamed rows in the beginning. We therefor need to set the header at row 17, this is done by setting `header=17`
- To better be able to see what each column means when accessing it through a browser, the header is repeated after 40 teams are listed, and before the next 40 teams are listed. We have to skip these rows using the `skiprows` parameter, and defining the indices of the rows to skip. Notice that the indices changes non-intuitively since we start the header at 17.
- The Team names columns contains digits that we are not interested in. To make sure teams are the same for both years, we remove anything not in the English alphabet from that column.

1.2 Histogram of Adjusted Defence for 5 Conferences

We construct the function `Adjusted_defense` which takes the given conference, and the year we are interested in and returns a `Numpy` array with the scores for the Adjusted Defense Efficiency. We use this function for each conference we want, and plot all of the five conferences together in one histogram. We chose a bar-stacked plot, to better be able to visualise it. The bins range from 85 to 115 with a width of 5 for all conferences. We obtain the plot seen in the Fig 1.

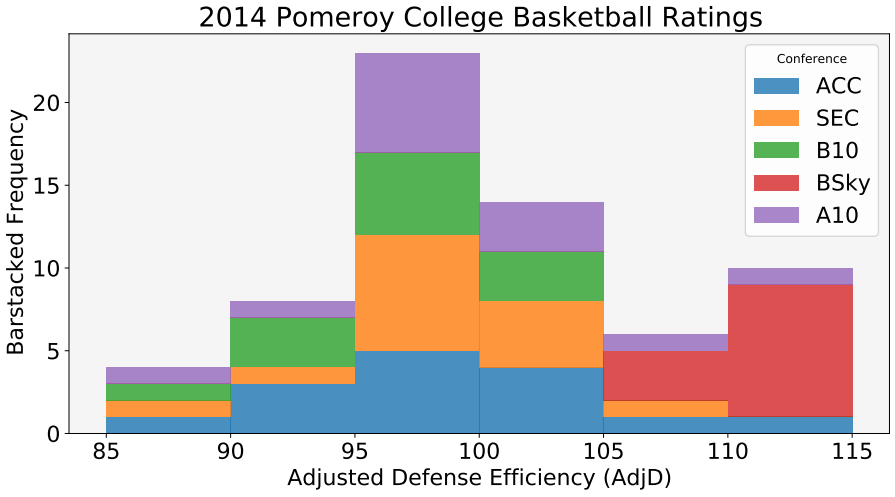

Figure 1: Histograms displaying the adjusted defense efficiency score for 5 conferences.

2 Exercise 2

Now we will look to data from both 2009 and 2014. First we investigate whether teams that compete both years are always in the same conference. This is not true for 78 teams, their conference will be denoted in this part as "Moved Conf". We construct thrre functions to help us solve this exercise:

- `extract_team_names` takes a conference and a year and returns a list with all teams meeting this requirement.
- `match_team_names` takes to list of teams (sometimes from the above function) and returns a list of the teams occuring in both lists.
- `Adjusted_offense_diff` takes a list of team names and returns their adjusted offense score from 2009, the difference in adjusted offense score (2014-2009) and a list of the conference they participated in.

2.1 Producing AdjO Graphs

We use `extract_team_names` and `match_team_names` to find the teams that competed in the same conference both years. We look to the 5 conferences stated in the assignment. We extract the difference in AdjO between the years and the AdjO for 2009 with the `Adjusted_offense_diff`, and we can now plot the results, labeling the data according to the conference. This is seen in Fig 3. We do the same with the rest

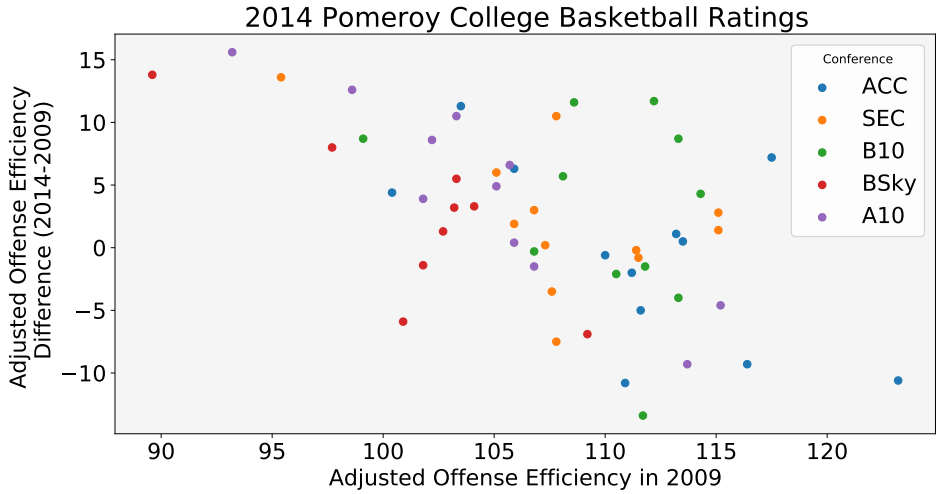

Figure 2: The difference in AdjO for one team between the years 2014 and 2009 as a function of the score in 2009. Team entries are labeled according to conference.

of the teams, those not competing in the 5 given conferences. This results in the plot displayed in Fig **??**.

2.2 Calculating Mean and Median of AdjO

We now calculate the mean and median for the difference in score between the two years. We sort this by conference. For the teams competing in the wanted 5 conferences the results are displayed below in Fig 4. The mean and medians of teams not competing in the wanted 5 conferences are displayed in Fig 5.

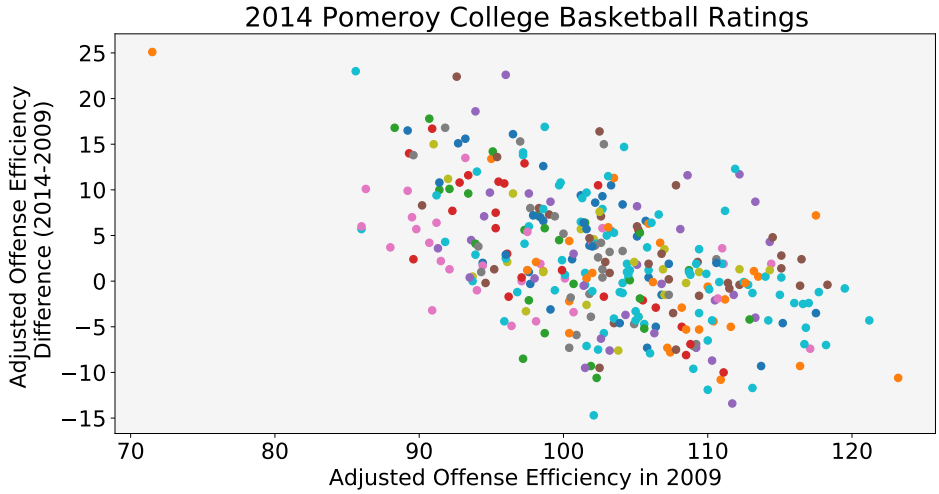

Figure 3: The difference in AdjO for one team between the years 2014 and 2009 as a function of the score in 2009. Team entries are labelled according to conference.

```
Conf: ACC        mean = -0.625   median = -0.050
Conf: SEC        mean = 2.283    median = 1.650
Conf: B10        mean = 2.673    median = 4.300
Conf: BSky       mean = 2.322    median = 3.200
Conf: A10        mean = 4.336    median = 4.900
```

Figure 4: The mean and the median of the difference in AdjO for teams competing in the same conference. The 5 given conferences are displayed here.

```
Conf: AE              mean = -0.800   median = -0.450
Conf: ASun            mean = 3.289    median = 2.400
Conf: B12             mean = 2.850    median = 0.850
Conf: BE              mean = 1.143    median = 1.900
Conf: BSth            mean = 4.200    median = 3.350
Conf: BW              mean = 1.713    median = 0.800
Conf: CAA             mean = 4.150    median = 6.900
Conf: CUSA            mean = -2.550   median = -3.650
Conf: Horz            mean = 2.075    median = 2.300
Conf: Ivy             mean = 7.137    median = 9.000
Conf: MAAC            mean = 4.511    median = 4.500
Conf: MAC             mean = 3.983    median = 3.400
Conf: MEAC            mean = 2.991    median = 2.200
Conf: MVC             mean = 2.644    median = 2.600
Conf: MWC             mean = 1.100    median = 1.150
Conf: Moved Conf      mean = 1.237    median = 0.450
Conf: NEC             mean = 2.711    median = 2.400
Conf: OVC             mean = 0.830    median = 0.600
Conf: Pat             mean = 7.587    median = 7.700
Conf: SB              mean = 0.943    median = 0.400
Conf: SC              mean = -0.045   median = 0.300
Conf: SWAC            mean = 2.950    median = 2.700
Conf: Slnd            mean = 3.067    median = 1.000
Conf: Sum             mean = -0.000   median = 1.200
Conf: WAC             mean = 2.250    median = 2.250
Conf: WCC             mean = 6.638    median = 6.350
Conf: ind             mean = 25.100   median = 25.100
```

Figure 5: The mean and the median of the difference in AdjO for teams competing in the same conference. The teams NOT competing in the 5 given conferences are displayed here.

3 Exercise 3

Exercise 3 repeats the calculation made in (1) and (2), with another conference added to the wanted conferences list. We use the same functions as before but with the new list and obtain the following

results.

We plot the Adjusted Defense Scores of all of the six conferences together in one histogram. We chose a bar-stacked plot, to better be able to visualise it. The bins range from 85 to 115 with a width of 5 for all conferences. We obtain the plot seen in the Fig 1.

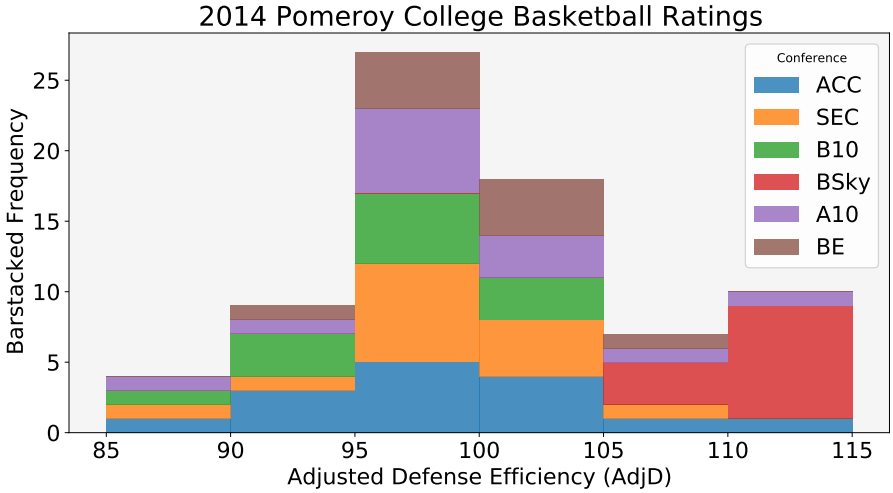

Figure 6: Histograms displaying the adjusted defense efficiency score for 6 conferences.

We look to the 6 conferences stated in the assignment. We extract the difference in AdjO between the years and the AdjO for 2009 with the `Adjusted_offense_diff`, and we can now plot the results, labeling the data according to the conference. This is seen in Fig 7.

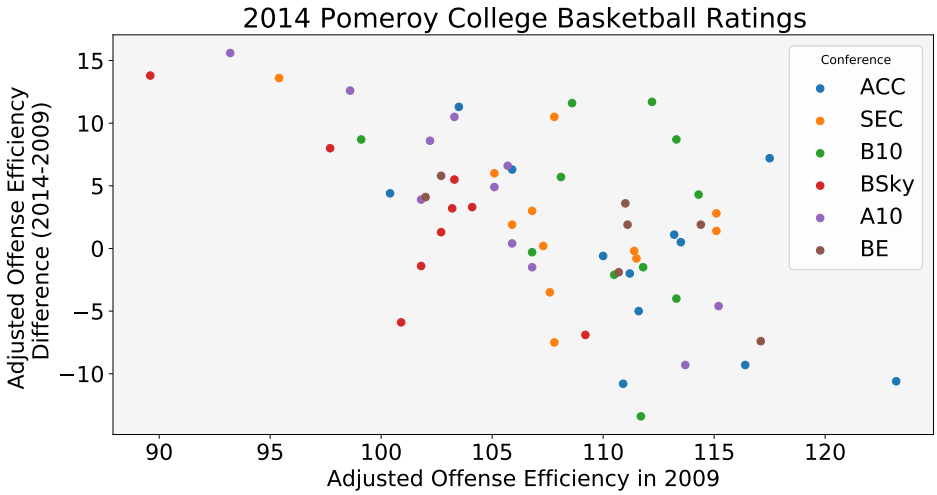

Figure 7: The difference in AdjO for one team between the years 2014 and 2009 as a function of the score in 2009. Team entries are labeled according to conference.

The mean and medians are also found in similar ways. The values for the six wanted conferences are

displayed in Fig 8 and the values for the rest of the conferences and the teams that moved conferences are displayed in Fig. 9.

```
Conf: ACC        mean = -0.625   median = -0.050
Conf: SEC        mean = 2.283    median = 1.650
Conf: B10        mean = 2.673    median = 4.300
Conf: BSky       mean = 2.322    median = 3.200
Conf: A10        mean = 4.336    median = 4.900
Conf: BE         mean = 1.143    median = 1.900
```

Figure 8: The mean and the median of the difference in AdjO for teams competing in the same conference. The 6 given conferences are displayed here.

```
Conf: AE               mean = -0.800   median = -0.450
Conf: ASun             mean = 3.289    median = 2.400
Conf: B12              mean = 2.850    median = 0.850
Conf: BSth             mean = 4.200    median = 3.350
Conf: BW               mean = 1.713    median = 0.800
Conf: CAA              mean = 4.150    median = 6.900
Conf: CUSA             mean = -2.550   median = -3.650
Conf: Horz             mean = 2.075    median = 2.300
Conf: Ivy              mean = 7.137    median = 9.000
Conf: MAAC             mean = 4.511    median = 4.500
Conf: MAC              mean = 3.983    median = 3.400
Conf: MEAC             mean = 2.991    median = 2.200
Conf: MVC              mean = 2.644    median = 2.600
Conf: MWC              mean = 1.100    median = 1.150
Conf: Moved Conf       mean = 1.237    median = 0.450
Conf: NEC              mean = 2.711    median = 2.400
Conf: OVC              mean = 0.830    median = 0.600
Conf: Pat              mean = 7.587    median = 7.700
Conf: SB               mean = 0.943    median = 0.400
Conf: SC               mean = -0.045   median = 0.300
Conf: SWAC             mean = 2.950    median = 2.700
Conf: Slnd             mean = 3.067    median = 1.000
Conf: Sum              mean = -0.000   median = 1.200
Conf: WAC              mean = 2.250    median = 2.250
Conf: WCC              mean = 6.638    median = 6.350
Conf: ind              mean = 25.100   median = 25.100
```

Figure 9: The mean and the median of the difference in AdjO for teams competing in the same conference. The conferences not in the 6 given conferences are displayed here.

4 Exercise 4: Authors

4.1 Extracting the text from the PDF

The text is extracted directly from the PDF-file linked in the problem-set description [1]. To scrape the PDF from text, we use the library `pdfplumber`[2]. To extract (almost) only the authors, we define a bounding box for each page telling the program where on the page to scan text from. There are authors up to p. 11 ind the PDF. The first page, has a header and therefor a peculiar bounding box. Pages 2 through 10 have the same bounding box, while the references begin on p. 11 and therefor this page also has a special bounding box. All bounding boxes are found from trial and error. Joining the strings of text from each page, we create the string `main_string` which contains all the extracted text.

[1] `http://www.nbi.dk/~koskinen/Teaching/AdvancedMethodsInAppliedStatistics2018/data/authors-acknowledgements-v5.pdf` [Accessed: 14.09, 10/02/2022]

[2] `https://github.com/jsvine/pdfplumber`

4.2 Cleaning the text, so it only contains authors

The `main_string` contains various characters that are not part of the author names, we therefor define the function `clean_string` which handles all non-author-characters except double commas and a few extra spaces. It follows the order:

- All "AND" words in the end of research groups are removed with a regular expression.
- All digits are removed with string operations.
- Parenthesis and their contents, the names of the research groups, are replaced with commas, using another regular expression.
- New line symbols are removed with string operations.
- The following special characters are removed with string operations: [´,‘,⋆,˜,¨,˘,„,ˇ,’,§,‡,†,-,ˆ,˙], and Ł is changed to an L. Notice that since the special characters were defined manually, we might have missed some.

This gives us a new string `clean_main_string`. This is made into a list of strings, using the commas as separators. In a for loop with string operations empty elements are removed (arises from double commas) along with extra spaces in the front.

4.3 Finding unique authors

Finding the number of unique authors is relatively straight forward using `Numpy.unique`. We find that there are 3612 authors total, 3513 unique author names and thus 99 repeated author names.

4.4 Alphabetizing

Since we have to alphabetize by last name and the first initial(s), we first need to manipulate the authors list slightly. We put the last name first, by searching for the last dot (dots go after each initial) in each author name, and then placing everything after (the last name) first in the string. We add a space here, and then add everything before the last dot (the initials) to the beginning of the string . Sometimes this leaves and extra space in the beginning of the string, we remove those at the end. As for the alphabetizing, this is straight forward using the built in python function `sorted`. Since we have an even number of authors, there are two authors in the middle. They are: LI C.K. and LI B.

There are a few considerations we have not accounted for here, but should be mentioned.

- We ignore hyphen when alphabetizing, so we will remove those completely by replacing them with nothing, i.e. removing them. I'm not sure how, for example "J.-D. FOURNIER" should be alphabetized, but here we sort it as if "-D" is just a "D". For compound last names, they are sorted correctly but are hard to read since the two names originally separated by comma are put together into one.

- A few authors have spaces in their last name, and not only a single name, examples are "VAN DER HORST", "S. DI PACE", "R. CHERKAOUI EL MOURSLI" and people with "Jr." or "Sr." in their name. With the built algorithm all spaces in the last name are removed, and they are sorted as if there are no spaces.
- For names such as "Van Der X", "Di X" and "El X" they are sorted by the first letter read, and not by the main part of their last names as it is sometimes done, depending on style.