

Total: 3+4+3+0.5 = 10.5 points

Problem Set #1

Advanced Methods in Applied Statistics



“Having four is preferable to just one.” - Cyan Jo

Thanks for making me
smile. A stressful PhD
student like me
definitely need this. :)

Cyan Yong Ho Jo

14th February 2024

SCROUNGING

To scrape data on past or current NCAA basketball leagues from Ken Pomeroy's website, interacting with a web browser is essential. I plan to try two methods: using either *JavaScript* or *Python*'s *requests* package. Each method has its advantages. Using *JavaScript* requires the additional tool *Node.js*, but data querying is more intuitive because *JavaScript* is inherently web-oriented, being event-driven and bi-directional. Furthermore, I can automate this process by calling it from my *Python* code for data analysis using the *subprocess.run()* function, which allows the execution of external files. Similarly, using *Python* necessitates installing the *requests* package, but it offers simplicity and ease of use. Regardless of the method, the data will be stored and managed in a subdirectory of my workspace.

Finding the URL pattern

Ken Pomeroy's website features a simple layout. The homepage, found at <https://kenpom.com/>, presents data from the latest NCAA basketball season. By appending a query string with a specific year to the main URL, you can access data for that particular year. For instance, if I were to scrap the content of the year 2014, I simply need to add a PHP query at the end of the main URL: <https://kenpom.com/index.php?y=2014>.

Extract HTML

Before starting the scraping process, it is beneficial to investigate the web page's structure by examining its source. I prefer using *Chrome* by *Google* because it is the most widely used web browser among web developers. This allows me to confirm the hierarchical structure and identify data stored within tabular containers. See Figure1.

The screenshot shows the Google Chrome developer tools open over a webpage titled "2014 Pomeroy College Basketball Ratings". The page displays a table of basketball ratings for various teams. A red box highlights the table element in the source code. Arrows point from this highlighted element to the corresponding node in the DOM tree and the "Elements" panel, illustrating how the tabular data is identified and managed within the page's structure.

Figure 1: "In *Chrome*'s Developer Mode, the source is analysed, and visual indicators show how each part of the source corresponds to specific elements on the page. The tabular data I identified (highlighted in a red box) are delineated by shading."

Parsing HTML files

The extracted HTML files contain both redundant data and the data I need for this project. For this purpose, I require another web scraping package called *BeautifulSoup*. `.find('table')` method in *BeautifulSoup* allows me to extract the data stored in a table, which is precisely what I need.

The header of the table extracted by *BeautifulSoup* has several undesirable features. Firstly, the header is not confined to a single row; it spans two rows, which complicates data processing. I need to modify the header to fit within a single row. Secondly, each header item contains unnecessary strings. For instance, a header item labeled ‘Rk’ is presented with additional, extraneous characters.

Unnamed: 0_level_0	Rk	Unnamed: 0_level_2	Rk	Unnamed: 0_level_8
_level_4	Rk	Unnamed: 0_level_6	Rk	Unnamed: 0_level_12
Rk	Unnamed: 0_level_10	Rk	Unnamed: 0_level_12	Rk
Unnamed: 0_level_14	Rk	Unnamed: 0_level_16	Rk	

I planned to solve these two problems by removing the current header and assigning a new one written by myself. For some columns, there is a tailing column that has an adjusted value of the original column. I chose to tag them with `enh` postfix.

```
HEADER = [ 'Rk' , 'Team' , 'Conf' , 'W' , 'L' , 'AdjEM' , 'AdjO' , 'AdjO_enh' , 'AdjD' , 'AdjD_enh' , 'AdjT' , 'AdjT_enh' , 'Luck' , 'Luck_enh' , 'AdjEM_SoS' , 'AdjEM_enh_SoS' , 'OppO_SoS' , 'OppO_enh_SoS' , 'OppD_SoS' , 'OppD_enh_SoS' , 'AdjEM_NCSOS' , 'AdjEM_enh_NCSOS' ]
# newly defined header.
```

The third issue I encountered was the format of the values of the wins and losses. They are combined in a string having ‘(W)-(L)’ form so it needed to be separated into two columns ‘(W)’ and ‘(L)’. Finally, I need to convert the numbers into a numerical format so that *Python* interpreter can recognise them as numbers. The initial format of the extracted data was string.

MATPLOTLIB FORMAT

The selection of color combinations in any plot is essential for ensuring visibility and comprehension. I have decided to introduce a preset for the *Matplotlib* package, which will establish the size and thickness of the lines, both axes, ticks, and font sizes for labels, titles, and legends. Additionally, I will set a cyclic property for markers and colors for the plot by using `plt.cycler()` function. I utilised colour searching and generating tool from this web-site: *Colors*. It shows patches of colours I picked and I can check the hexadecimal code of each colour and its name.

EX1

The histograms depicted below illustrate the AdjD values for teams belonging to the ACC, SEC, B10, BSky, or A10 conferences, utilizing data from 2014. When plotting a bar-type histogram, I encountered an issue with overlapping bars. To resolve this, I adjusted the layout by separating the bars and placing each one within the same bin but positioned next to each other, similar to the arrangement seen in Fig2 overlapping conference data. For the sake of visibility and recognition, I tried removing either the lines(in Fig3) or the bars(in Fig4).

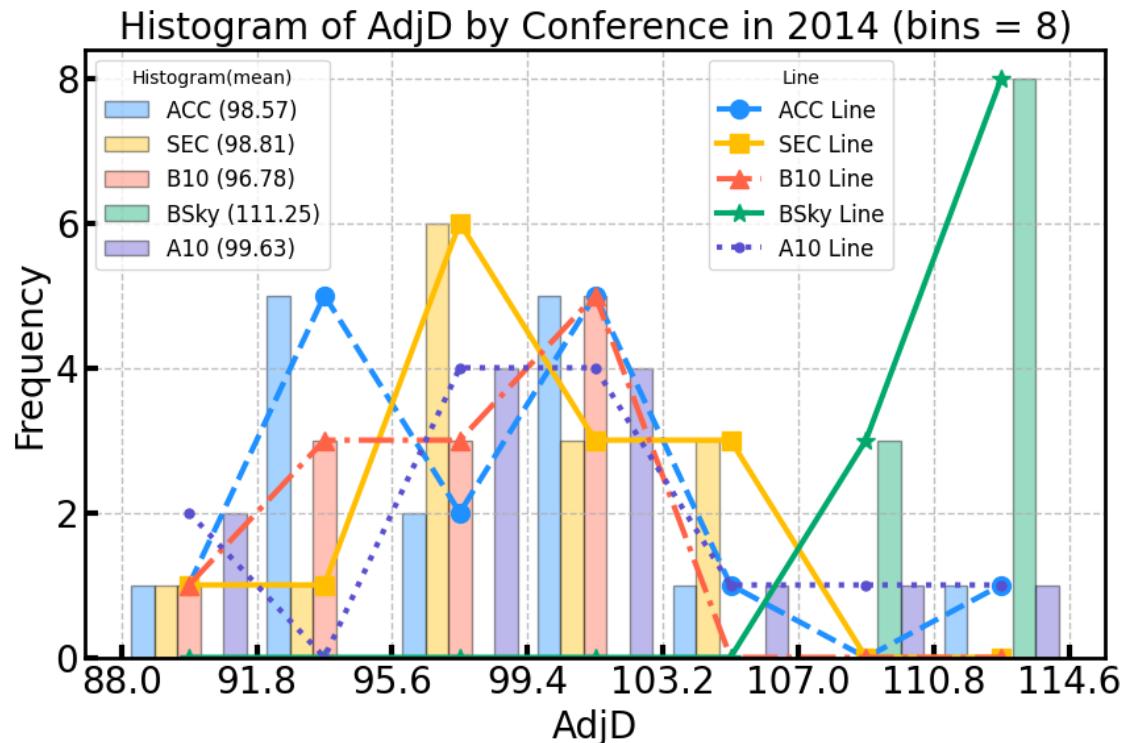


Figure 2

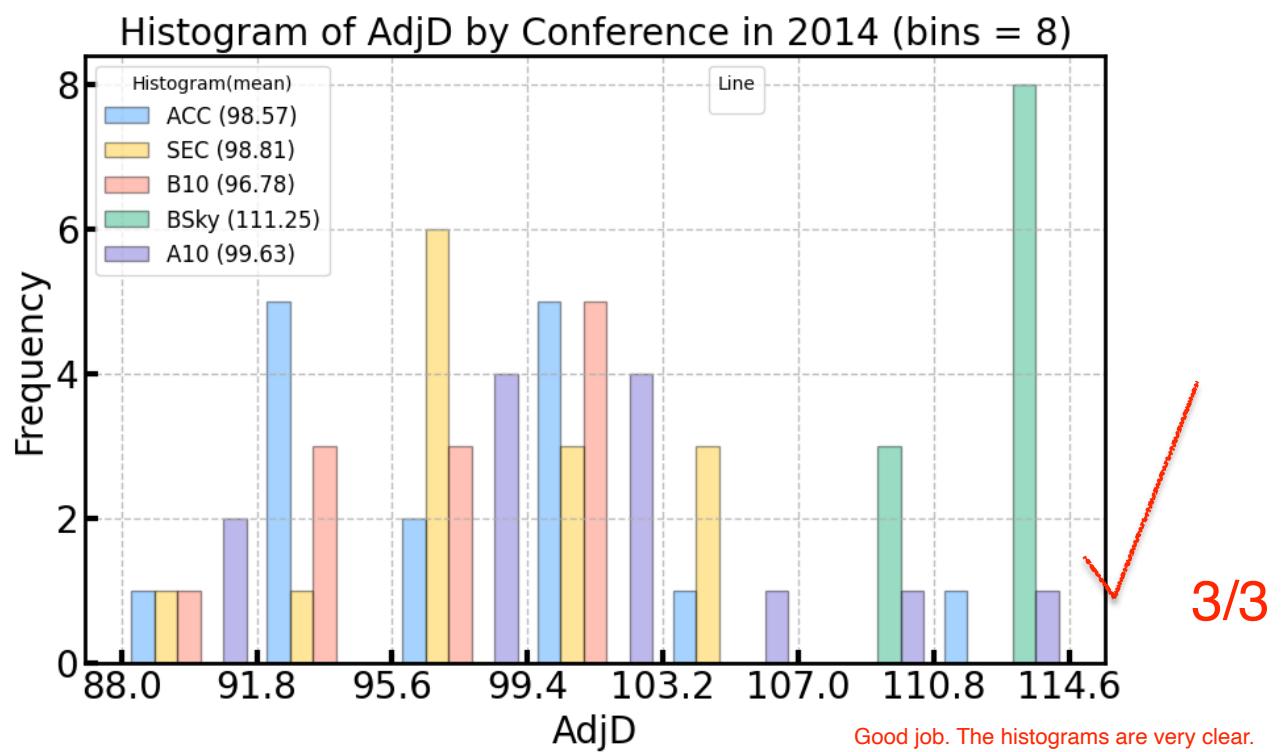


Figure 3

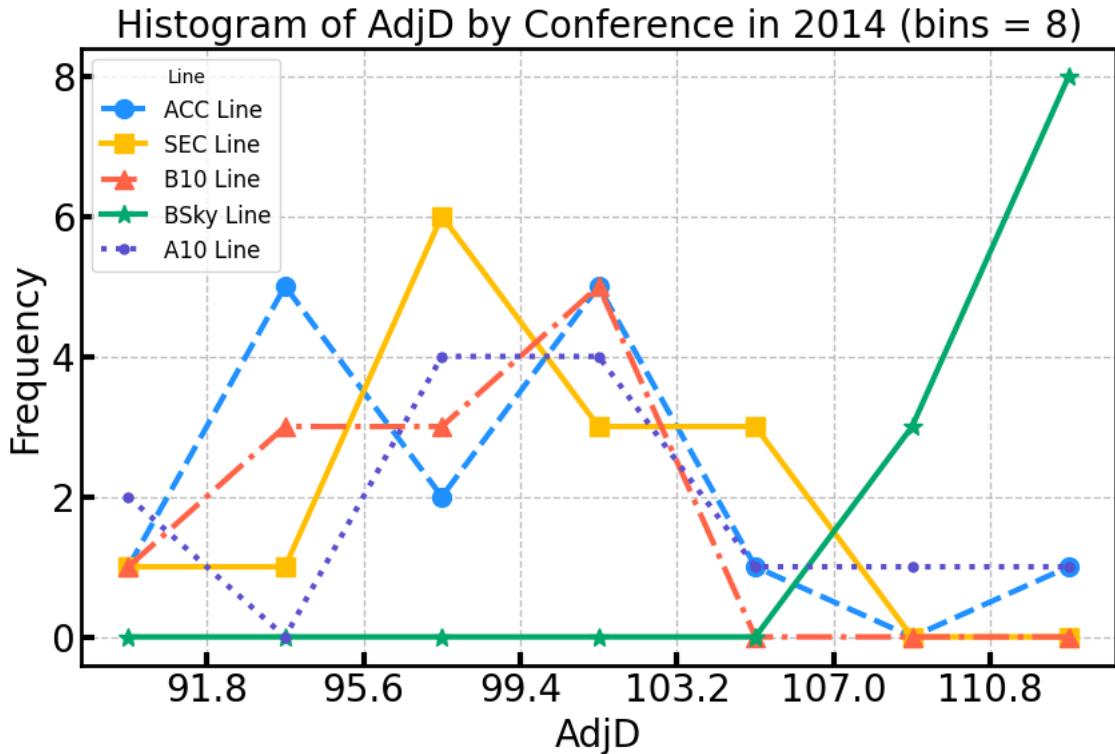


Figure 4

EX2

The scatter graph depicted in Fig5 showcases the fluctuation in AdjO values across teams affiliated with the ACC, SEC, B10, BSky, or A10 conferences between 2009 and 2014, about their AdjO values from 2009. The methodology employed for computing the discrepancies in AdjO might seem rudimentary. However, alterations in conference compositions throughout the five years have resulted in significant outliers in AdjO values, nearing the 100 mark. To address this issue, I opted to eliminate teams with available data for either the 2009 or 2014 periods. I checked this by plotting the difference of each team on Fig6. Recalling ACC is the conference whose teams marked the lowest values of AdjD in 2014 overall, the teams in the conference have not experienced a notable increase in AdjO in general.

The mean and median are shown in TableI. ACC is the only conference that has undergone decreased value in AdjO in 2014 compared to 2009.

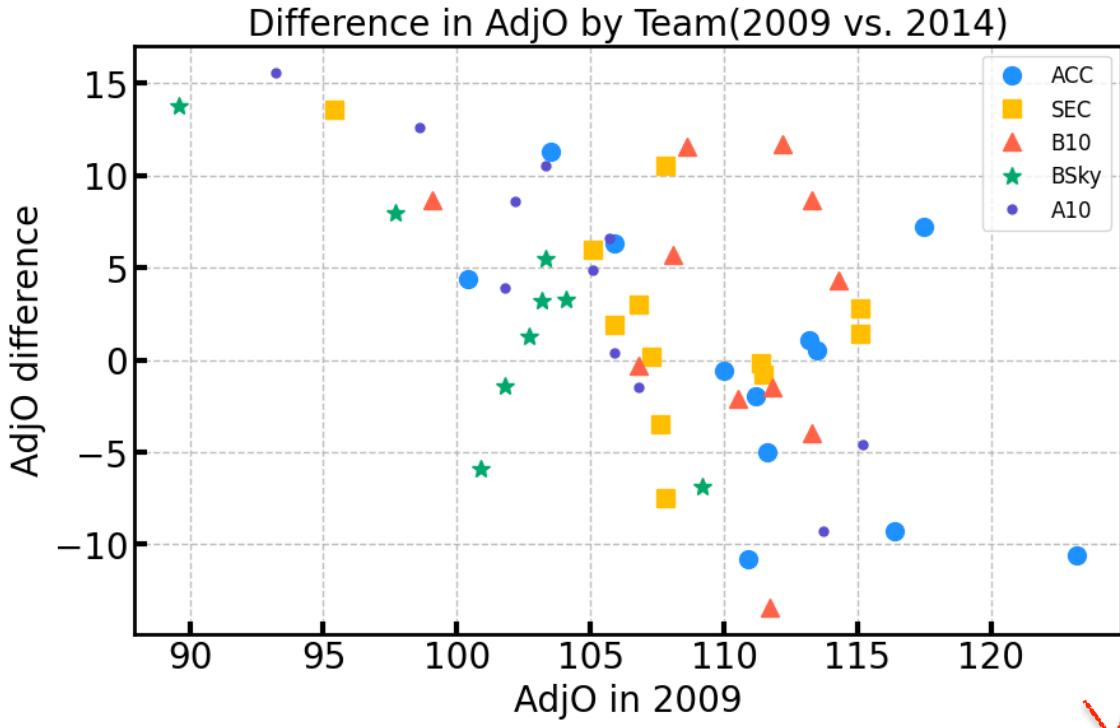


Figure 5

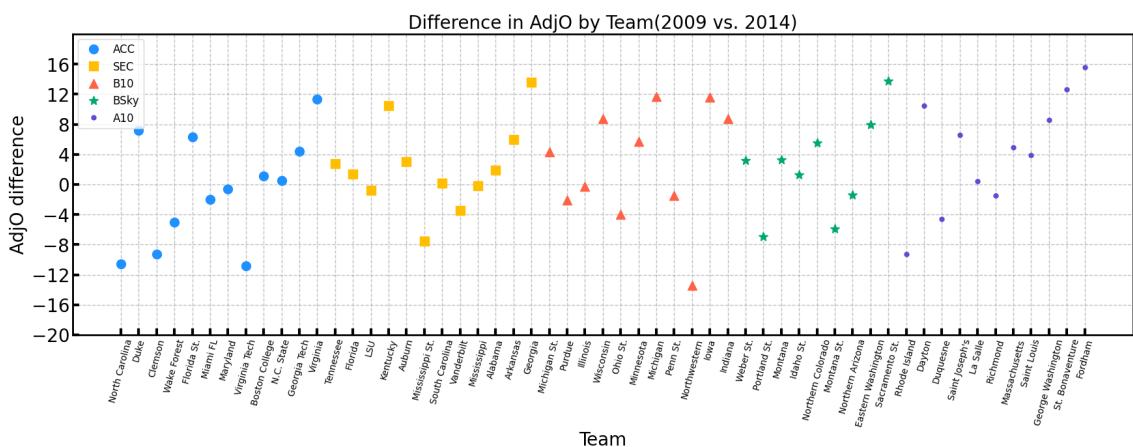


Figure 6

Conference	Mean	Median
ACC	-0.625	-0.050
SEC	2.283	1.650
B10	2.673	4.300
BSky	2.322	3.200
A10	4.336	4.900
Others	2.595	1.900

4/4

Table I: Mean and median of difference in AdjO values for five different conferences

Well done! The scatter plot and the mean/median calculations are correct.

EX3

Below are histograms of AdjD for the six conferences in 2014 Fig7, Fig8, and Fig9. AdjD of BE seems to have a similar trend as B10.

The mean and median of the difference in AdjO between 2014 and 2009 are shown in TableII.

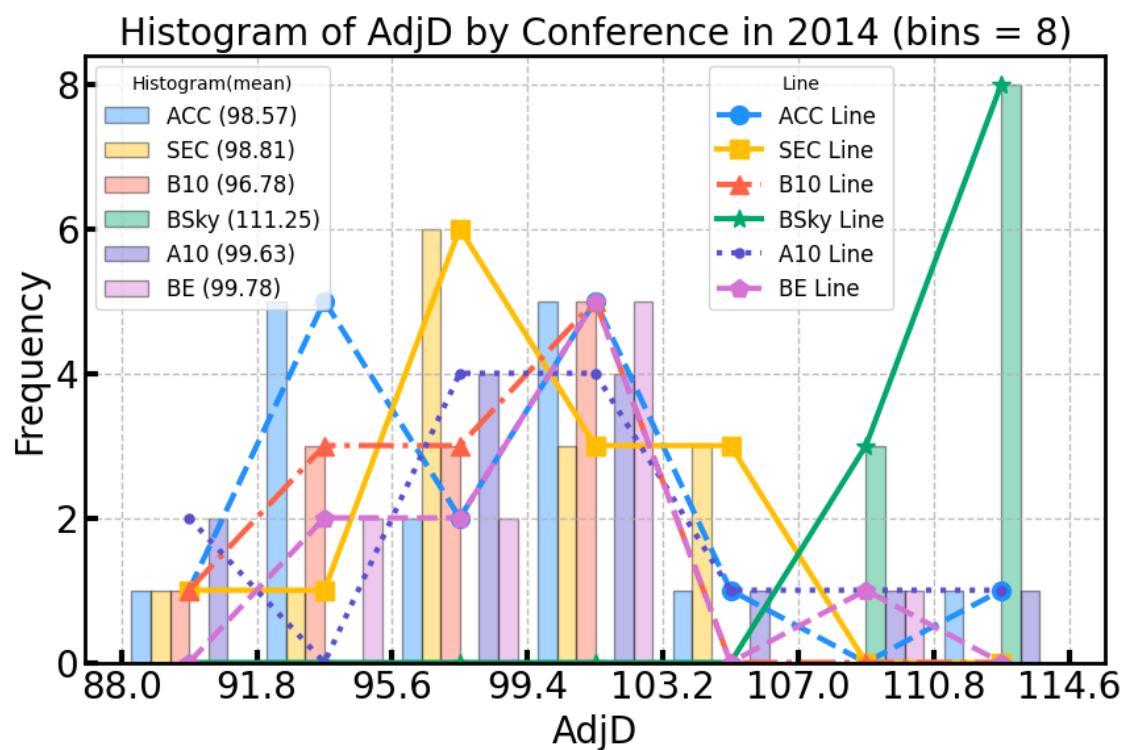


Figure 7

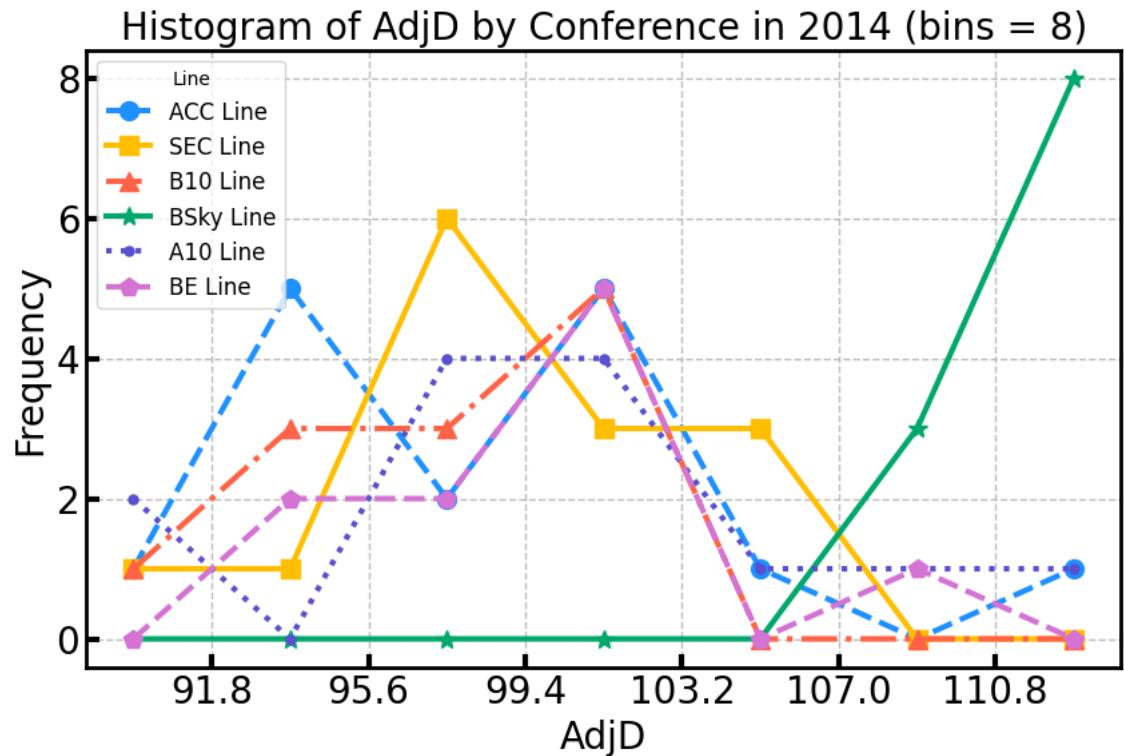


Figure 8

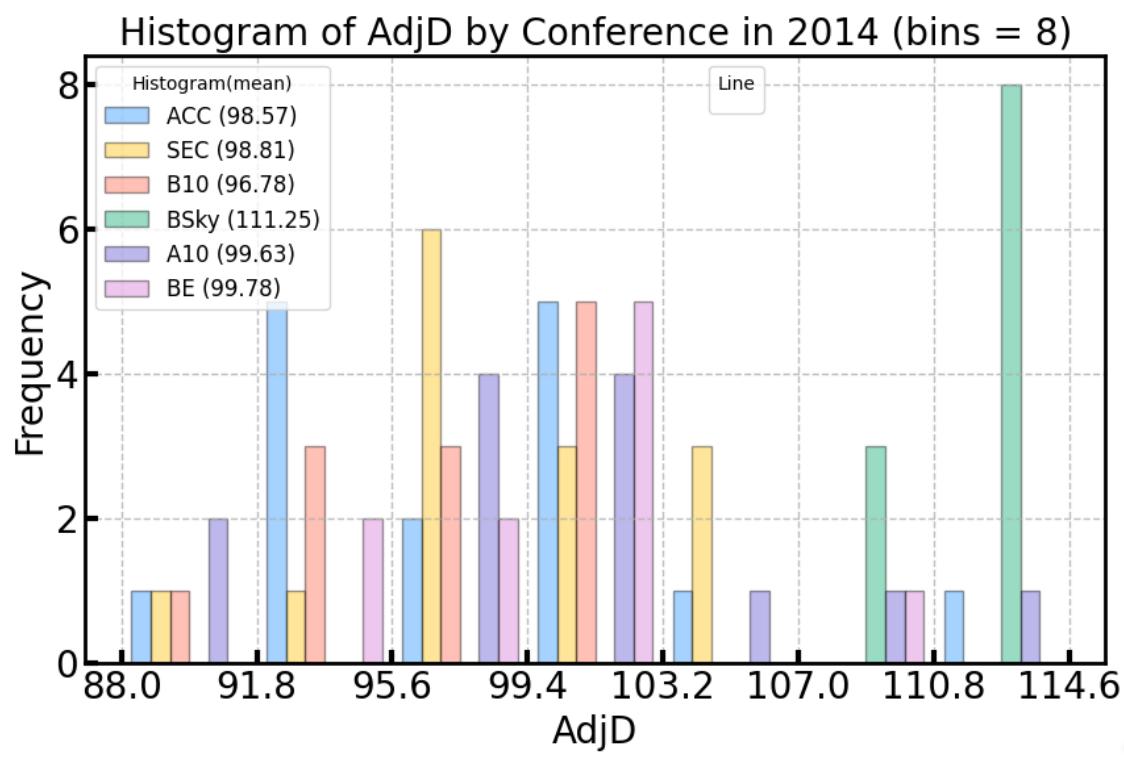


Figure 9

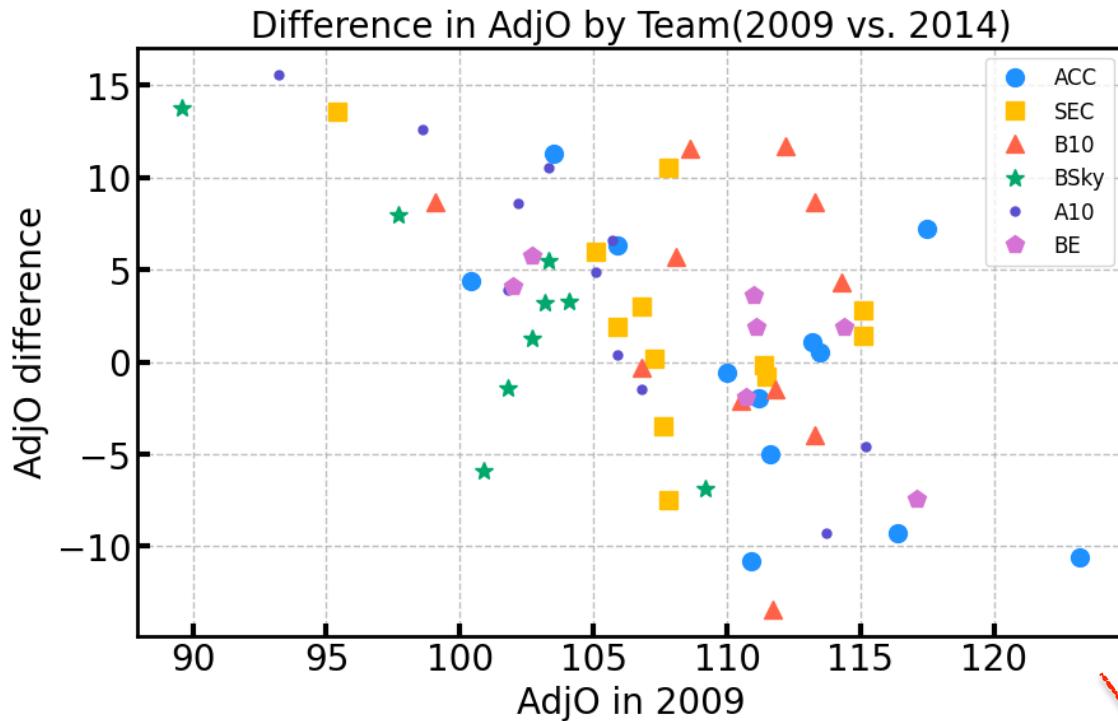


Figure 10

Conference	Mean	Median
ACC	-0.625	-0.050
SEC	2.283	1.650
B10	2.673	4.300
BSky	2.322	3.200
A10	4.336	4.900
BE	1.143	1.900
Others	2.624	1.850

3/3

Table II: Mean and median of difference in AdjO values for six different conferences

Well done. For question 3, the point is to make sure you can extend your codes from the previous exercises easily to more conferences.

EX4

I read the .pdf file and parsed it using javascript. It was a trick to handle non-ASCII characters like the letters with diacritics. The patterns I found are these.

- each name ends with a comma
- redundant line exists
- some lines has more than one names
- replacing special characters introduce additional line

The name in the middle of the alphabetically ordered list I found is **K. GILL**.

0.5/1

The mid author is slightly off, but point is given for the effort, plus the difference is acceptable. Good job.

Unfortunately you are missing the number of unique authors as requested by the question, so 0.5 point has to be deducted. (I tried to find it in your jupyter notebook, but it is not there as well.)