IS5126 HANDS-ON WITH APPLIED ANALYTICS

GUIDED PROJECT - TEAM02

A0228167Y JIN YINING A0191517A Han Bihui
A0228610M ZENG LIJIE A0212492L zhang runtian

Part A: Web Scraping

1.1 Identify one or more starting URL to crawl and explain your scraper's workflow.

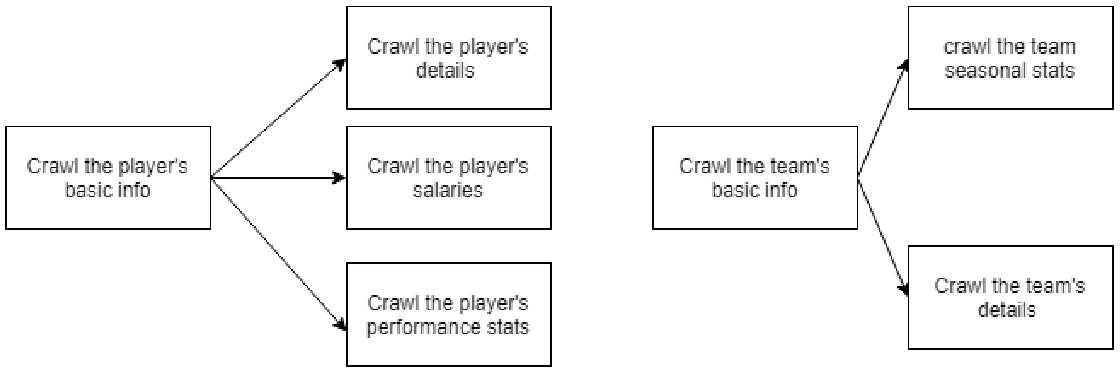
For players:

- https://www.basketball-reference.com/players/{a to z}/
- https://www.basketball-reference.com/players/{a to z}/{player_code}.html

For teams:

- https://www.basketball-reference.com/teams/
- https://www.basketball-reference.com/teams/{team_code}.html

Below is the main work flow:


Crawl the player's basic info
Crawl the player's details
Crawl the player's salaries
Crawl the player's performance stats
Crawl the team's basic info
crawl the team seasonal stats
Crawl the team's details

Firstly, we have used the library of request and tool of *Beautifulsoup4* to crawl the basic player's profile information such as name, position, height/weight, team and DOB. The previous operation is able to get all players' profile pages who are active during the 2009/2010 season to the 2019/2020 season and to crawl the player's details like recruiting rank, draft team and experience and their seasonal performance statistics.

Secondly, we crawled the player salaries in players profile pages (same as the players pages crawled before) through the tool of selenium to overcome the website's anti-crawler mechanism.

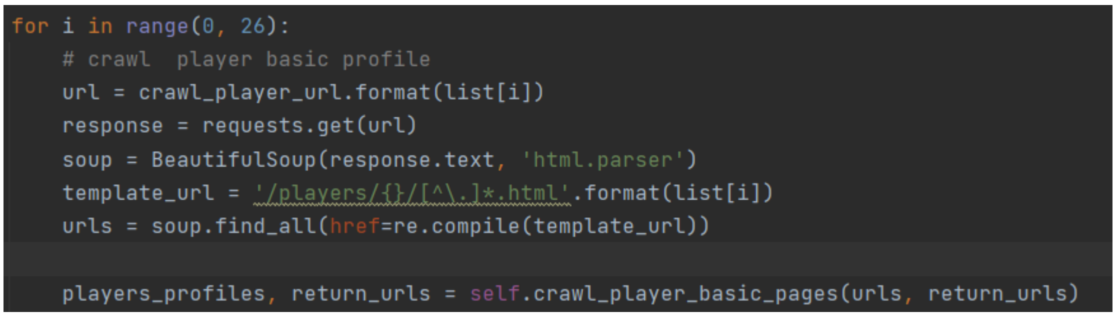
Thirdly, the library of request and tool of Beautifulsoup4 is used to crawl the basic team information including name, playoff appearance, championship and etc. The previous operation is able to get all team's detail pages and to crawl the team details like location, seasons played, record and their seasonal statistics from the 2009/2010 season to the 2019/2020 season.

1.2 Identify the links to follow using either BeautifulSoup 4, or CSS/XPath and provide your code to do so. (3 points)

Since my source code is too long, I only put some examples here.

1.2.1 For players:

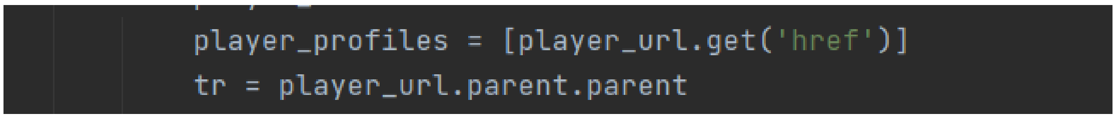
Data:['url', 'name', 'from', 'to', 'pos', 'ht', 'wt', 'dob', 'college', 'active']



```
for i in range(0, 26):
    # crawl  player basic profile
    url = crawl_player_url.format(list[i])
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    template_url = '/players/{}/[^\.]*.html'.format(list[i])
    urls = soup.find_all(href=re.compile(template_url))

    players_profiles, return_urls = self.crawl_player_basic_pages(urls, return_urls)
```

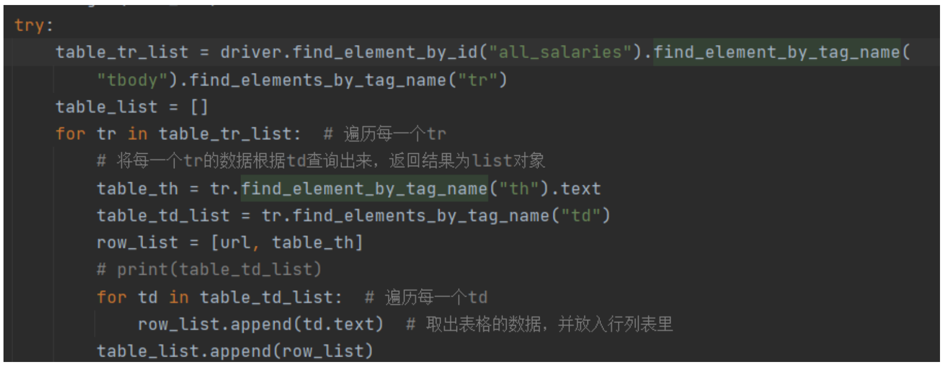
Regular expressions and findAll function have been used to find all player’s urls. And url.parent.parent or url.parent.parent.parent are used to find the table’s path.



```
            player_profiles = [player_url.get('href')]
            tr = player_url.parent.parent
```

Then, I separate the table head named th which includes name and table data named td which includes other infos. By the way, I also identified the active players whose tag named Strong. The players’ detail urls have been crawled using previous operations to get more details including player rank, draft_team, experience.

Data : ['player_url', 'season', 'team', 'lg', 'salary']

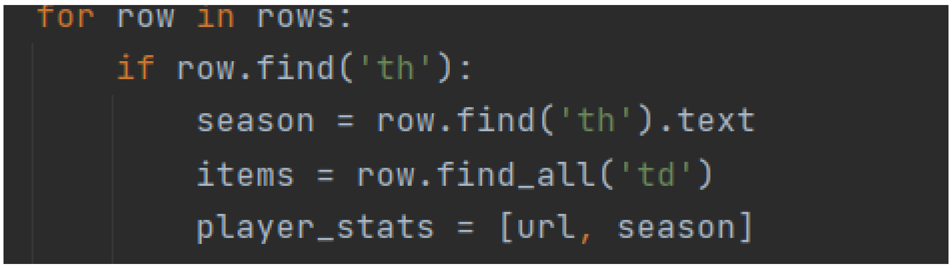


```
try:
    table_tr_list = driver.find_element_by_id("all_salaries").find_element_by_tag_name(
        "tbody").find_elements_by_tag_name("tr")
    table_list = []
    for tr in table_tr_list:  # 遍历每一个tr
        # 将每一个tr的数据根据td查询出来，返回结果为list对象
        table_th = tr.find_element_by_tag_name("th").text
        table_td_list = tr.find_elements_by_tag_name("td")
        row_list = [url, table_th]
        # print(table_td_list)
        for td in table_td_list:  # 遍历每一个td
            row_list.append(td.text)  # 取出表格的数据，并放入行列表里
        table_list.append(row_list)
```

To crawl the player’s salaries, I used selenium which is different from bf4. So, I find the element by id and tags to target the salary table and then traverse th and td in the table.

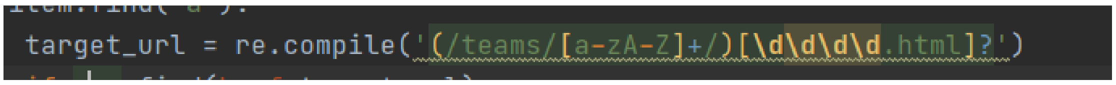
Data: ['player_url', 'Season', 'Age', 'Tm', 'Tm_url', 'Lg', 'Pos', 'G', 'GS', 'MP', 'FG', 'FGA', 'FG%', '3P','3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', BLK', 'TOV', 'PF', 'PTS']

Firstly, I use the find to target the Per Game Table and I traverse th and td in the table.



```
for row in rows:
    if row.find('th'):
        season = row.find('th').text
        items = row.find_all('td')
        player_stats = [url, season]
```

And, for the team_url, I used a regular expression to target its position and successfully extract its url.

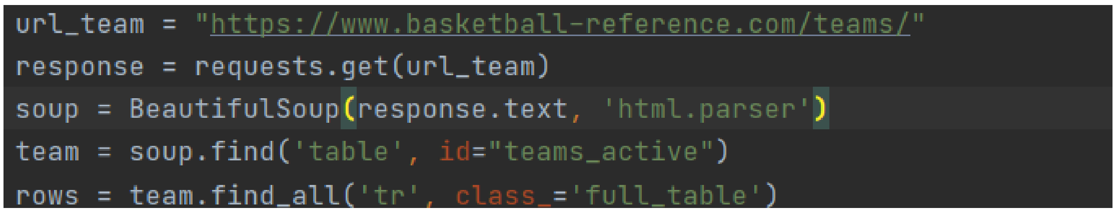


```
target_url = re.compile('(/teams/[a-zA-Z]+/)[\d\d\d\d.html]?')
```

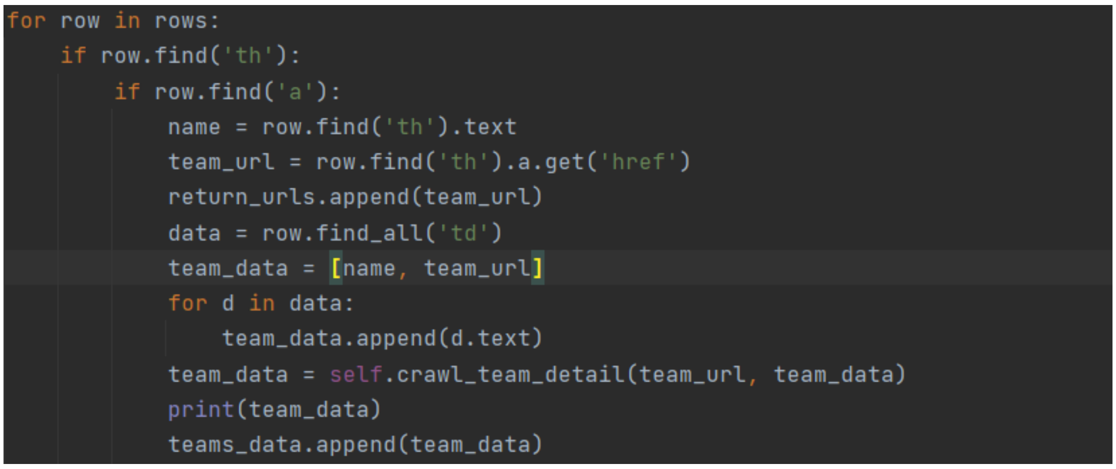
For teams:

Data:['name', 'url', 'lg', 'from', 'to', 'years', 'games', 'wins', 'losses', 'wlpercent', 'playoffs', 'div', 'conf', 'champ']

Firstly, I use the find to target the team’s active table.

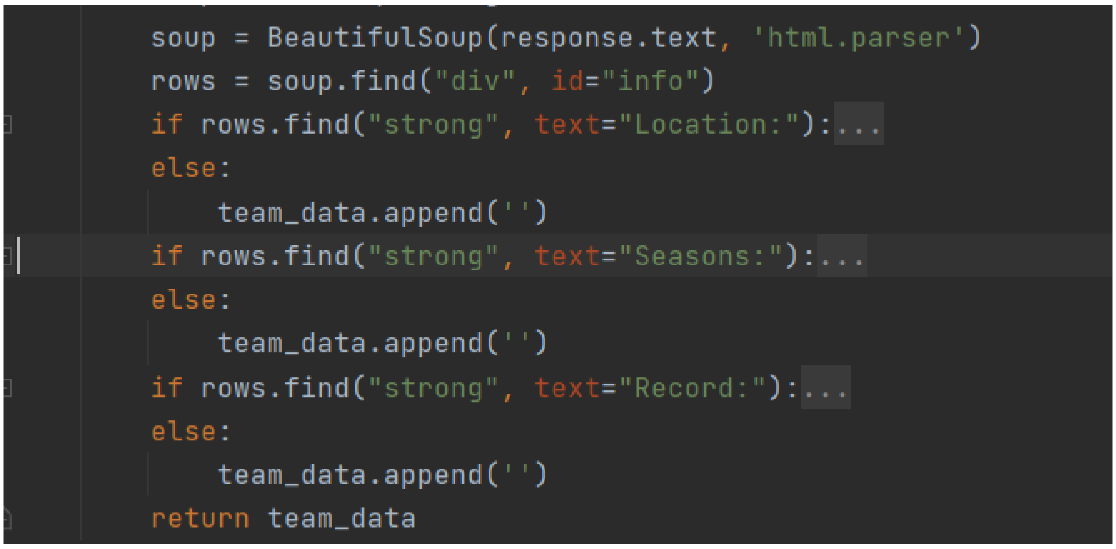

```
url_team = "https://www.basketball-reference.com/teams/"
response = requests.get(url_team)
soup = BeautifulSoup(response.text, 'html.parser')
team = soup.find('table', id="teams_active")
rows = team.find_all('tr', class_='full_table')
```

And, I traverse all rows in the table to get such data.


```
for row in rows:
    if row.find('th'):
        if row.find('a'):
            name = row.find('th').text
            team_url = row.find('th').a.get('href')
            return_urls.append(team_url)
            data = row.find_all('td')
            team_data = [name, team_url]
            for d in data:
                team_data.append(d.text)
            team_data = self.crawl_team_detail(team_url, team_data)
            print(team_data)
            teams_data.append(team_data)
```

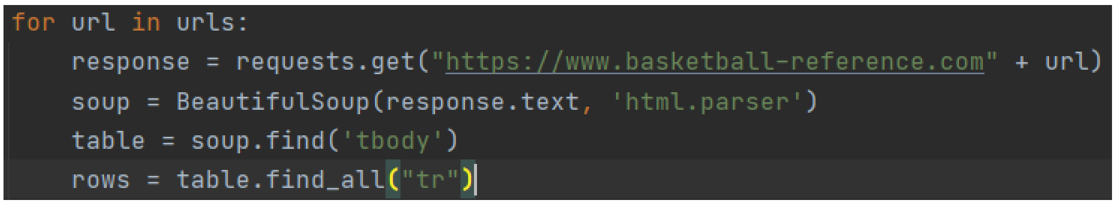
Data: ['location', 'seasons_played', 'record']

I crawled these data same as player's detail. I use the team url which I crawled in previous operation to target the page and then traverse such infos I needed.


```
        soup = BeautifulSoup(response.text, 'html.parser')
        rows = soup.find("div", id="info")
        if rows.find("strong", text="Location:"):...
        else:
            team_data.append('')
        if rows.find("strong", text="Seasons:"):...
        else:
            team_data.append('')
        if rows.find("strong", text="Record:"):...
        else:
            team_data.append('')
        return team_data
```

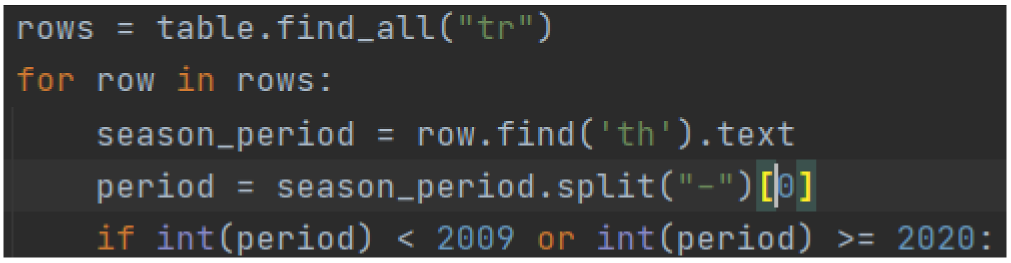
Data: ['url', 'Season', 'Lg', 'Team', 'W', 'L', 'W/L%', 'Finish', 'SRS', 'Pace', 'Rel_Pace', 'ORtg', 'Rel_ORtg', 'DRtg', 'Rel_DRtg', 'Playoffs', 'Coaches', 'Top_WS']:

Firstly, I used the url I crawled to target the seasonal performance table.



```python
for url in urls:
    response = requests.get("https://www.basketball-reference.com" + url)
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('tbody')
    rows = table.find_all("tr")
```

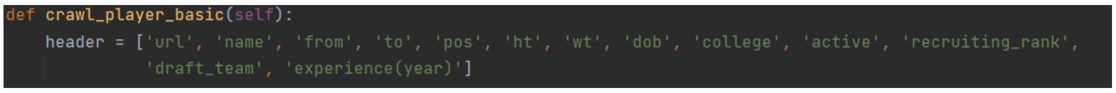
And, I traverse all th and td in tables to crawl the data. By the way, I also set a range to filter the seasonal data.



```python
rows = table.find_all("tr")
for row in rows:
    season_period = row.find('th').text
    period = season_period.split("-")[0]
    if int(period) < 2009 or int(period) >= 2020:
```

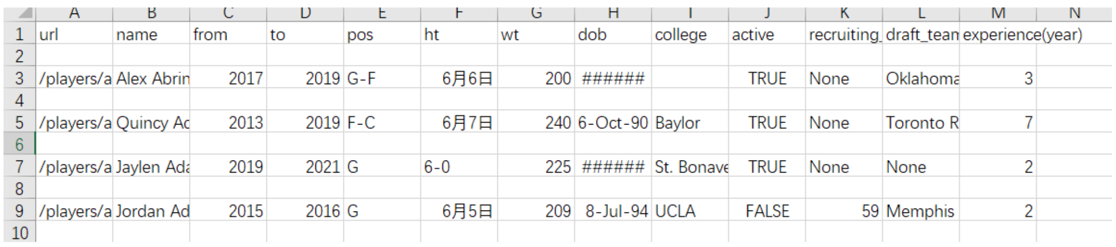
1.3 On each player's page, you should at least parse the following information: (10 points)

Basic player's profile information such as name, position, height/weight, team, birthday (age), recruiting rank, draft team, experience, etc.



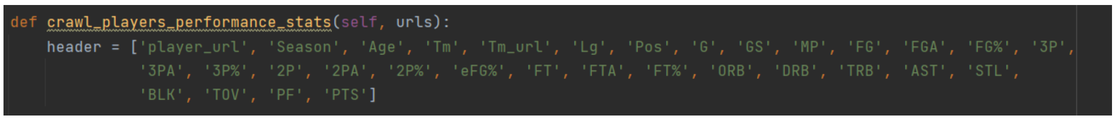
```python
def crawl_player_basic(self):
    header = ['url', 'name', 'from', 'to', 'pos', 'ht', 'wt', 'dob', 'college', 'active', 'recruiting_rank',
              'draft_team', 'experience(year)']
```

The details of data are in the attachment.



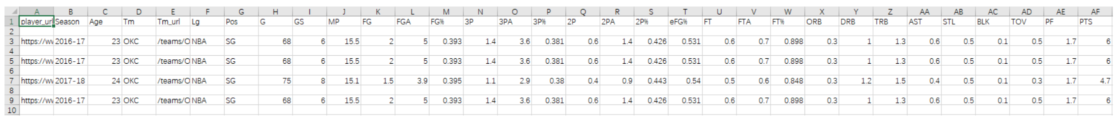
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	url	name	from	to	pos	ht	wt	dob	college	active	recruiting_	draft_tean	experience(year)	
2														
3	/players/a	Alex Abrin	2017	2019	G-F	6月6日	200	######		TRUE	None	Oklahoma	3	
4														
5	/players/a	Quincy Ac	2013	2019	F-C	6月7日	240	6-Oct-90	Baylor	TRUE	None	Toronto R	7	
6														
7	/players/a	Jaylen Ada	2019	2021	G	6-0	225	######	St. Bonave	TRUE	None	None	2	
8														
9	/players/a	Jordan Ad	2015	2016	G	6月5日	209	8-Jul-94	UCLA	FALSE	59	Memphis	2	
10														

a. Player's performance statistics.



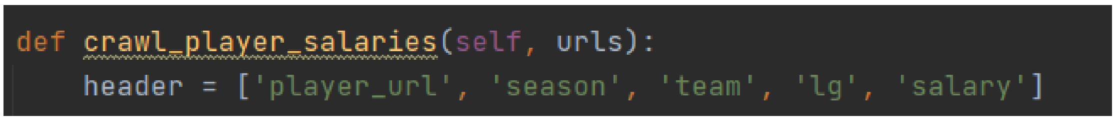
```python
def crawl_players_performance_stats(self, urls):
    header = ['player_url', 'Season', 'Age', 'Tm', 'Tm_url', 'Lg', 'Pos', 'G', 'GS', 'MP', 'FG', 'FGA', 'FG%', '3P',
              '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL',
              'BLK', 'TOV', 'PF', 'PTS']
```

The details of data are in the attachment.



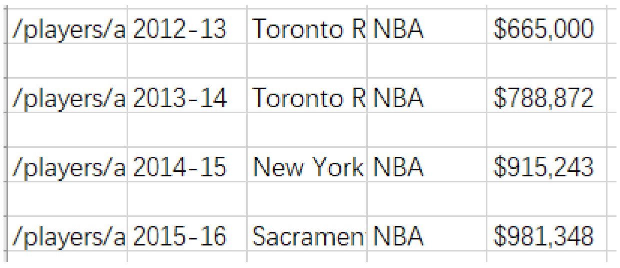
player_url	Season	Age	Tm	Tm_url	Lg	Pos	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	eFG%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
https://wv	2016-17	23	OKC	/teams/O	NBA	SG	68	6	15.5	2	5	0.393	1.4	3.6	0.381	0.6	1.4	0.426	0.531	0.6	0.7	0.898	0.3	1	1.3	0.6	0.5	0.1	0.5	1.7	6
https://wv	2016-17	23	OKC	/teams/O	NBA	SG	68	6	15.5	2	5	0.393	1.4	3.6	0.381	0.6	1.4	0.426	0.531	0.6	0.7	0.898	0.3	1	1.3	0.6	0.5	0.1	0.5	1.7	6
https://wv	2017-18	24	OKC	/teams/O	NBA	SG	75	8	15.1	1.5	3.9	0.395	1.1	2.9	0.38	0.4	0.9	0.443	0.54	0.5	0.6	0.848	0.3	1.2	1.5	0.4	0.5	0.1	0.3	1.7	4.7
https://wv	2016-17	23	OKC	/teams/O	NBA	SG	68	6	15.5	2	5	0.393	1.4	3.6	0.381	0.6	1.4	0.426	0.531	0.6	0.7	0.898	0.3	1	1.3	0.6	0.5	0.1	0.5	1.7	6

c. Player's salaries.



```
def crawl_player_salaries(self, urls):
    header = ['player_url', 'season', 'team', 'lg', 'salary']
```

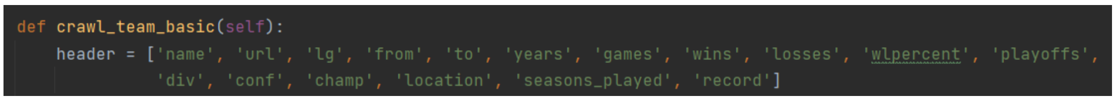
The details of data are in the attachment.



/players/a	2012-13	Toronto R	NBA	$665,000
/players/a	2013-14	Toronto R	NBA	$788,872
/players/a	2014-15	New York	NBA	$915,243
/players/a	2015-16	Sacramen	NBA	$981,348

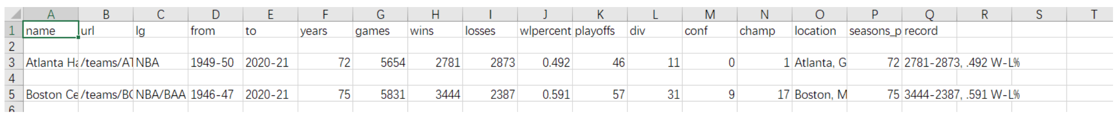
1.4 On each team (franchises) page, at least parse the following information: (10 points)

a. Basic team information (top panel on the page) including name, location, seasons played, record, playoff appearance, championship, etc.



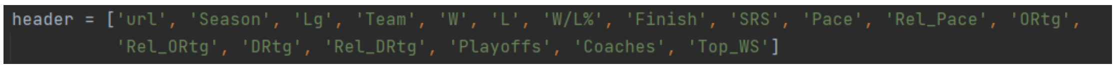
```
def crawl_team_basic(self):
    header = ['name', 'url', 'lg', 'from', 'to', 'years', 'games', 'wins', 'losses', 'wlpercent', 'playoffs',
              'div', 'conf', 'champ', 'location', 'seasons_played', 'record']
```

The details of data are in the attachment.



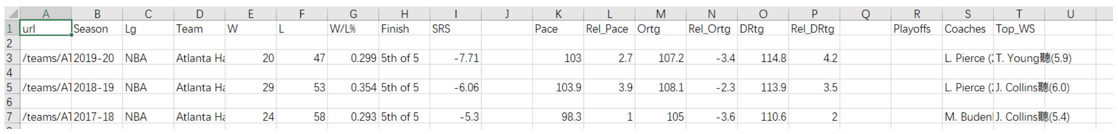
name	url	lg	from	to	years	games	wins	losses	wlpercent	playoffs	div	conf	champ	location	seasons_p	record
Atlanta Ha	/teams/A1	NBA	1949-50	2020-21	72	5654	2781	2873	0.492	46	11	0	1	Atlanta, G	72	2781-2873, .492 W-L%
Boston Ce	/teams/B(	NBA/BAA	1946-47	2020-21	75	5831	3444	2387	0.591	57	31	9	17	Boston, M	75	3444-2387, .591 W-L%

b. Team seasonal statistics.



```
header = ['url', 'Season', 'Lg', 'Team', 'W', 'L', 'W/L%', 'Finish', 'SRS', 'Pace', 'Rel_Pace', 'ORtg',
          'Rel_ORtg', 'DRtg', 'Rel_DRtg', 'Playoffs', 'Coaches', 'Top_WS']
```

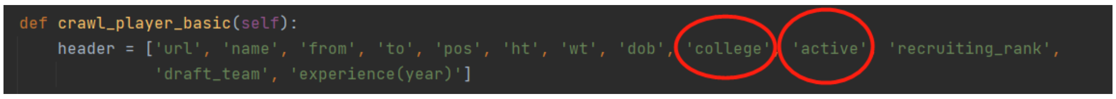
The details of data are in the attachment.



url	Season	Lg	Team	W	L	W/L%	Finish	SRS		Pace	Rel_Pace	Ortg	Rel_Ortg	DRtg	Rel_DRtg		Playoffs	Coaches	Top_WS
/teams/A1	2019-20	NBA	Atlanta Ha	20	47	0.299	5th of 5	-7.71		103	2.7	107.2	-3.4	114.8	4.2			L. Pierce (:	T. Young聽(5.9)
/teams/A1	2018-19	NBA	Atlanta Ha	29	53	0.354	5th of 5	-6.06		103.9	3.9	108.1	-2.3	113.9	3.5			L. Pierce (:	J. Collins聽(6.0)
/teams/A1	2017-18	NBA	Atlanta Ha	24	58	0.293	5th of 5	-5.3		98.3	1	105	-3.6	110.6	2			M. Buden	J. Collins聽(5.4)

1.5 Other information. What other raw data from the site might be helpful to build analytics for team owners? Please include your method, code, data and reasoning. (5 points)

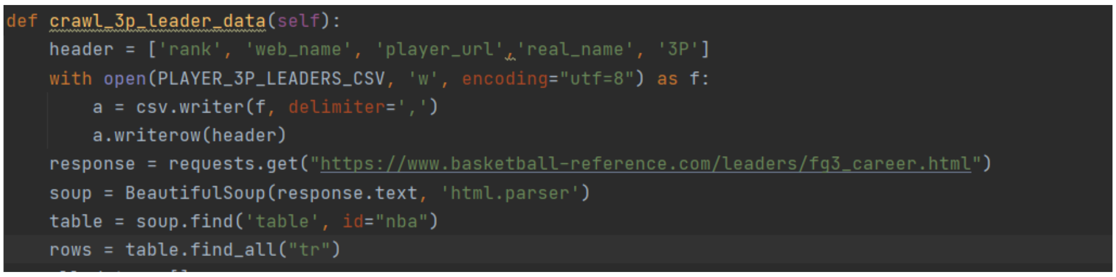
Beside the requirement, I crawled the player's college and whether he is an active player. Data are provided in the attachment with the table of players basic info.



```
def crawl_player_basic(self):
    header = ['url', 'name', 'from', 'to', 'pos', 'ht', 'wt', 'dob', 'college', 'active', 'recruiting_rank',
              'draft_team', 'experience(year)']
```

I chose college data because I think college is the NBA's one of largest talent delivery bases, so finding the data about colleges is necessary. And, I chose active players' data to distinguish between active and retired players, so as to compare whether players in different periods will have different changes in data due to the technological progress of basketball.

And, I also crawled top 250 NBA Records for 3-Pt Field Goal.



```
def crawl_3p_leader_data(self):
    header = ['rank', 'web_name', 'player_url','real_name', '3P']
    with open(PLAYER_3P_LEADERS_CSV, 'w', encoding="utf-8") as f:
        a = csv.writer(f, delimiter=',')
        a.writerow(header)
    response = requests.get("https://www.basketball-reference.com/leaders/fg3_career.html")
    soup = BeautifulSoup(response.text, 'html.parser')
    table = soup.find('table', id="nba")
    rows = table.find_all("tr")
```

I crawled these data the same as before. I found the record url to target the record page and then traverse the tables to crawl such infos I needed.

Due to the changes in the rules of the NBA , the 3-Pt Field Goal in the NBA is now a very advantageous means of scoring. For example, NBA stars such as Stephen Curry used the 3-Pt Field Goal to create the Warriors dynasty. As a team owner, I think learning more about the record of 3-Pt Field Goal might be helpful.

Part B: Data Preparation and Description

2.1 Put all the relevant data variables into one dataframe. Explain how you clean your dataset and transform your data variable if any and provide a data dictionary for all your variables.

First, we handle the datasets we scraped from the website. Change the column names to correct the false alignment problem. The core method is pd.DataFrame.merge().

At last, we get two integral table player.csv and team.csv, player.csv is the important one.

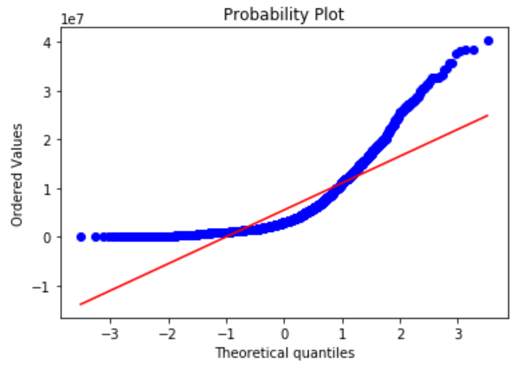
For the rest of the questions, please refer to Q2.ipynb.

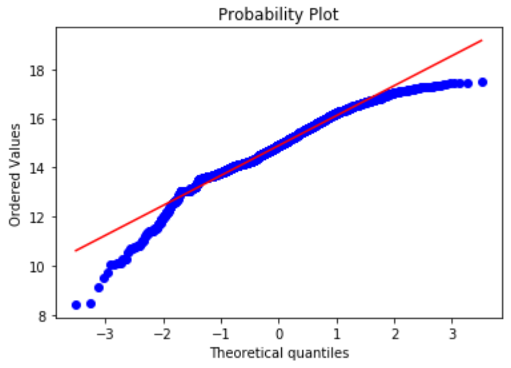
Part C: Data Analytics (50 points)

3.1 KMeans Modelling

3.1.1 Data Analysis:

Based on the most current season data, the salary Q-Q plot is shown on the bottom left, which is skewed toward the left. In order to remove this skewness, salary_log is calculated to fit the curve in the normal distribution and use salary throughout the modelling.

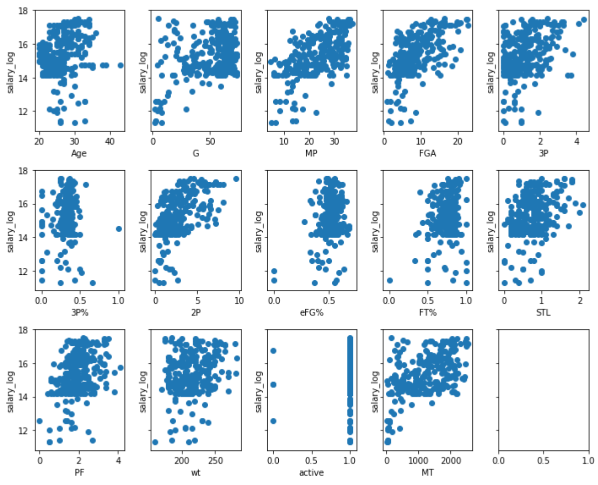




3.1.2 Feature Selection

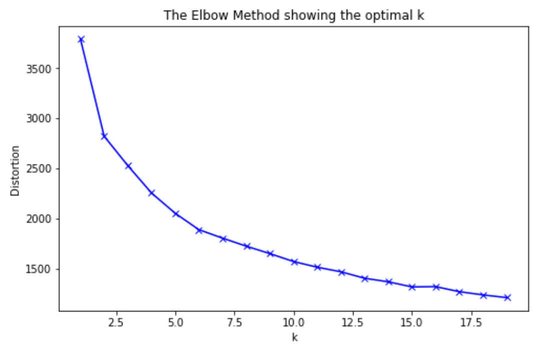
We have identified 30 features from the NBA website. In order to select suitable features for our modelling, backward elimination feature selection has been conducted with p-value limit set at 0.05. As such, there are 14 features left and the scatter plots between se are shown as below.

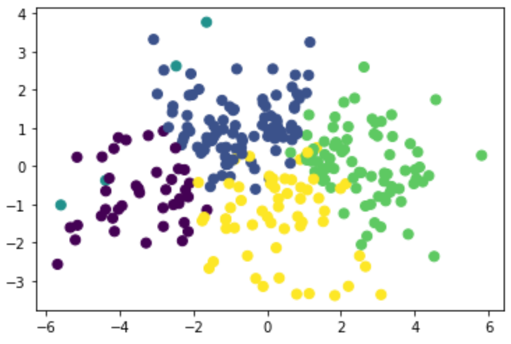
From the scatter plots, we have observed that MP (Minutes Played), FGA(Field Goals Attempted), 2P, and 3P show strong linear correlation to salary, which suggests that players with better performance will get higher salary. In addition, parameters including MT(Total Played minutes Per Season) and G(Games Played) also show good relationship to salary. Meanwhile, we have observed that lower salary will be paid to players with age larger than 26 year old.



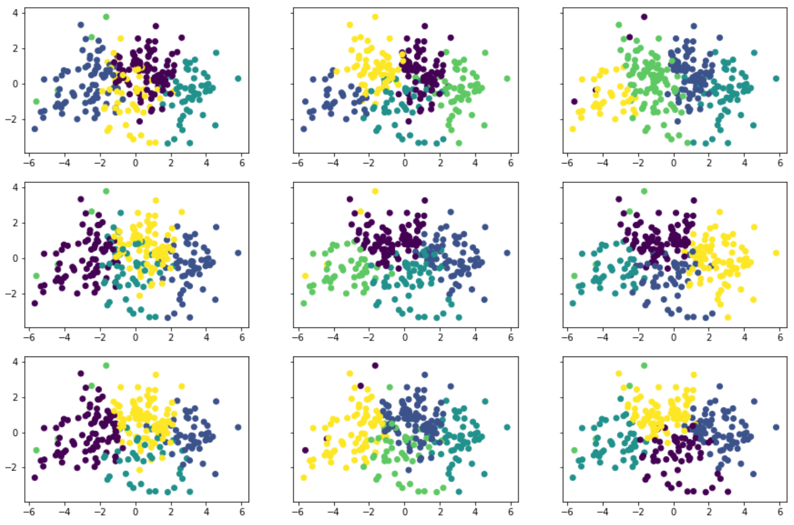
3.1.3 KMeans Modelling

Based on the elbow plot shown as below, we have decided to choose k=5 as the cluster number as the line starting to decline gradually from 5 onwards. In order to select a better centroid starting points***, init='k-means++'*** have been set which is to choose initial points which are as far apart from each other as possible. This method has been identified as one of the best centroid initialization strategies, which helps k-Means converge to the Global Minimum in just a few iterations [1]. The KMeans result using PCA visualization is shown as the figure on the bottom right.



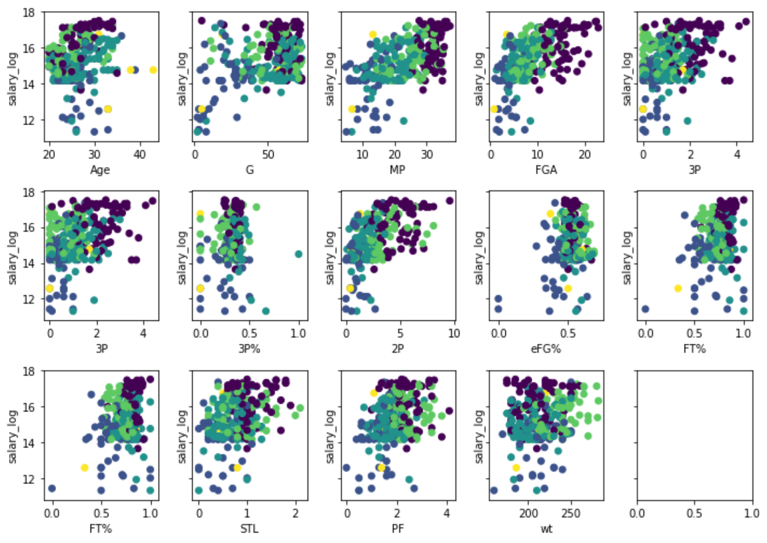


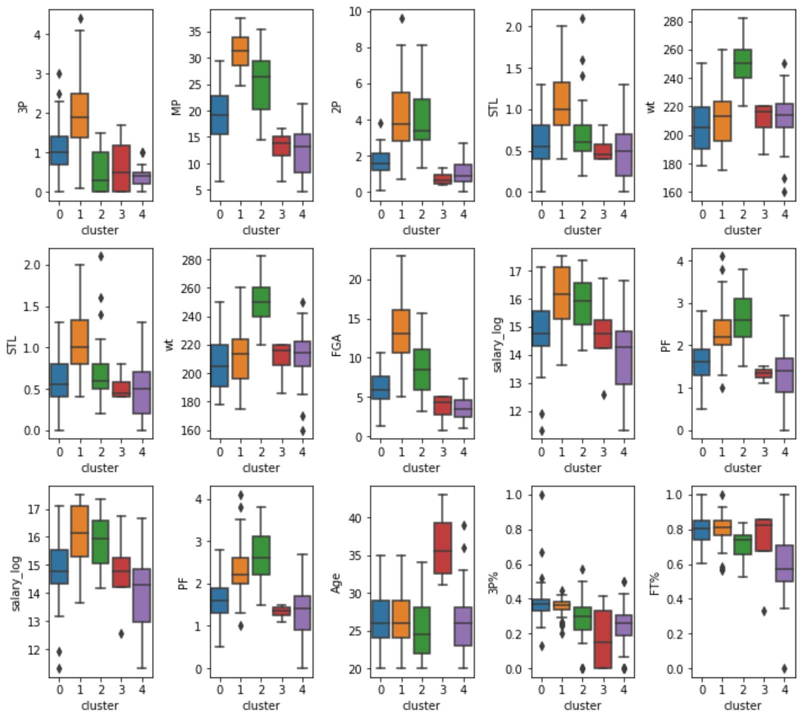
Meanwhile, we have also tried to initialize starting points using different random seeds. According to the plots shown, we have selected 9 random seeds [0, 24, 48, 72, 96, 10, 100, 300, 500] and observed that different random seeds end up with different clustering results.



3.1.4 Modelling Result Discussion

According to the KMeans clustering result, cluster 0 has the highest salary while cluster 1 and 4 have the relatively lower salaries. **Cluster 0** players have better performance than the players from the other clusters in terms of MP, 3P, 2P, FT% and eFG%, which are in line with our knowledge. Oppositely, cluster1 and cluster 4 players both receive lower salary due to the poorer performance in terms of MP, 3P, 2P, FT% and eFG% among all the players. This observation again supports that the player performance is the key element to determine the player salary. Interestingly, cluster4 players have significant higher ages(>30 year old). Although they have played more games than cluster1 player, their minutes played per game is relatively low. For cluster 2 and 3, players receive medium salaries in NBA.





3.2 Linear Regression

Regression Equation

log(salary) = 15 + 0.013* G-0.365*

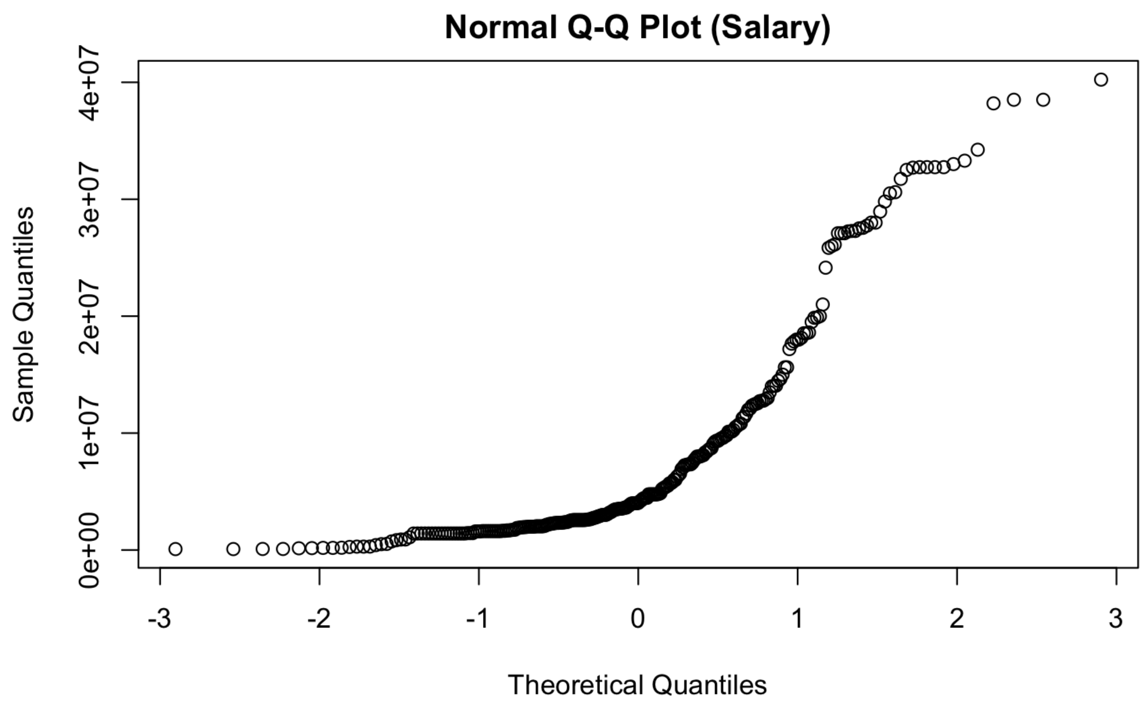
FG+0.232*PTS-0.188*FT+0.126*TRB-0.458*PF+0.146*TOV+0.062*BLK+

0.213*experience-0.102*Age

we observe that the regression model is deeply inclined towards the experience of a player, as it contributes a major factor in predicting a players salary.

On the contrary, MP, FG and FT was calculated as the least significant parameters through this model.

3.3 Model Justification

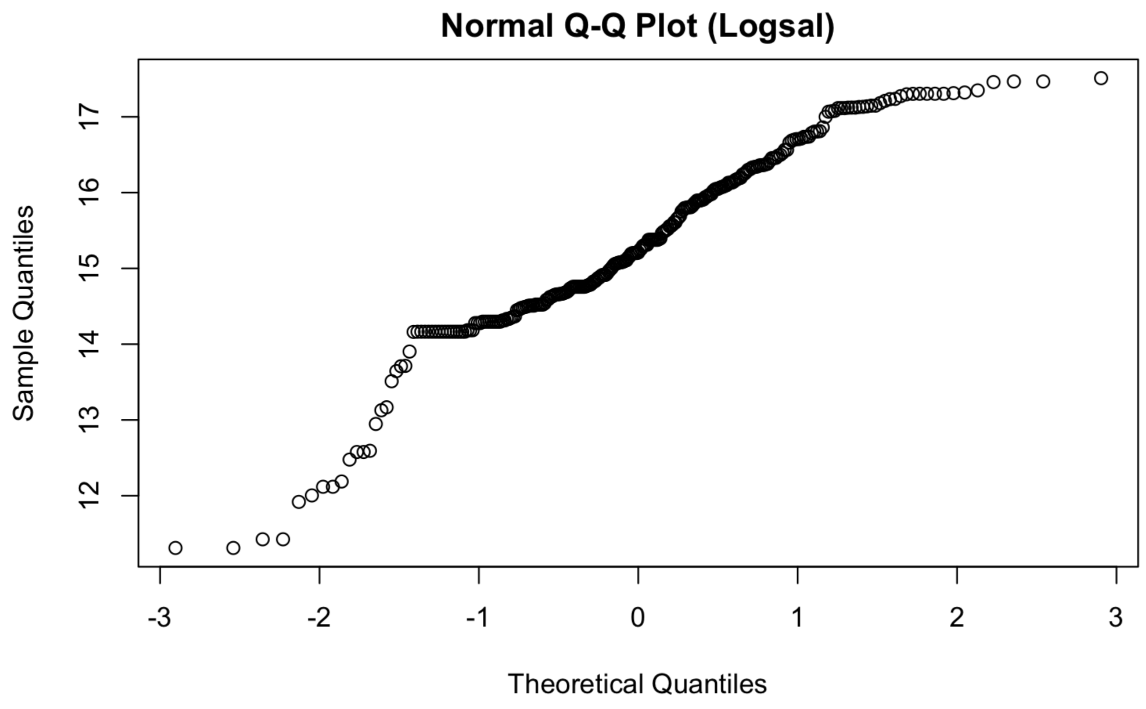


As we have illustrated above the line is not linear so it is concluded that heteroskedastic tendencies exist.

The conclusion is therefore that the assumption for constant variance is not satisfied.

In order to certify that the data is heteroskedastic, a Breusch-Pagan test is performed.

The Breusch-Pagan test generated a P-value below 0.05. Since the hypothesis for homoskedasticity is rejected it can be suggested that the data of the model is heteroskedastic.



By processing this method it was suggested to transform the response variable to log(salary)

```
Initial Model:
logSal ~ d.f$experience + d.f$FG + d.f$MP + d.f$PTS + d.f$G +
    d.f$TWOP + d.f$THTEEP + d.f$FT + d.f$TRB + d.f$PF + d.f$TOV +
    d.f$Pos_PF + d.f$Pos_PG + d.f$Pos_SF + d.f$Pos_SG

Final Model:
logSal ~ d.f$experience + d.f$FG + d.f$MP + d.f$PTS + d.f$G +
    d.f$TWOP + d.f$FT + d.f$TRB + d.f$PF + d.f$Pos_PF + d.f$Pos_PG +
    d.f$Pos_SG


          Step Df  Deviance Resid. Df Resid. Dev       AIC
1                                 255   182.8963 -74.55712
2 - d.f$THTEEP  1 0.2179901       256   183.1143 -76.23431
3 - d.f$Pos_SF  1 0.7281508       257   183.8424 -77.15882
4    - d.f$TOV  1 0.7914458       258   184.6339 -77.99466
```

stepwise AIC was performed to estimate if a single covariate can be excluded from the model.

For each covariate with a high p-value, a ΔAIC was calculated. where the Reduced model consisted of all variables except one and the Full model of all variables.

If the ΔAIC was negative the covariate was excluded from the final model. This was repeated stepwise until no ΔAIC was negative.

If I want to fit a multicollinearity, I would subtract the mean of the predictor variable from the value of the predictor variable.

Remove highly correlated predictors from the model. Because they provide redundant information, deleting them usually does not significantly reduce R^2. I will consider using stepwise regression, best subset regression, or data set expertise to remove these variables.

Use partial least squares or principal component analysis. These methods can reduce the number of predictors to a smaller set of uncorrelated components.

For example, a toy manufacturer wants to predict customer satisfaction. They include "strength" and "no breakage" as predictors in the regression model. The investigator can determine that these two variables have a strong negative correlation and a VIF greater than 5. Investigators can also use partial least squares or principal component analysis to use these related variables to create "durability" parts.

3.4 Panel Data Analysis

In the past ten seasons, many changes may have occurred in the career of basketball players. For example, a player may have moved to another team, or missed for several years due to an injury, and may return to the game in the following season. There may also be positions related to playing basketball. Moreover, with the increase in experience, the performance of the players may be correspondingly improved, resulting in an increase in salary. Similarly, this data is also very suitable for panel data analysis, because usually player data (personal data) is aggregated by season (time). The overall indicators of basketball players throughout the season can be roughly divided into two categories.

Time Invariant Variables included: Height, Weight, Playerid.

Certain parameters regarding the occupation of basketball players will always remain the same and will not change with the seasons.

Time Variant Variables included: ORB, DRB, BLK, TOV, PF,, AST, STL, PTS, experience,G, MP, THTEEP, TWOP

Other variables will be affected according to seasons and may have an influence on the salary of a basketball player.

Time Indicator include: Season

Fixed and random effects model:

Since experience was found to be a significant predictor of salary. so as experience increase , nab players' salaries will be increased.

The confidence interval of the fixed-effect model is drawn to predict the salary of the player based on the panel data analysis. All important salary predictors (for example, experience, or games played) do not contain 0 in their upper or lower limits. Therefore, the slope of salary forecast indicators makes a greater contribution to different salary differences.

Reference:

[1] Comparison of Initialization strategies for k-Means. (2021). Retrieved 7 March 2021, from https://medium.com/analytics-vidhya/comparison-of-initialization-strategies-for-k-means-d5ddd8b0350e