AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Prepared and Submitted by

NAME	ID
TARAFDER, BISHANATH	19-41827-3
AHAMED HASAN	20-41987-1

Course Name:Introduction To Data Science

Section:B

Project Title: Apply Web Scraping and Data Pre-processing

Submitted to

Dr. Akinul Islam Jony

Department of Computer Science

Faculty of Science and Technology,AIUB

Date of Submission

03-05-2023

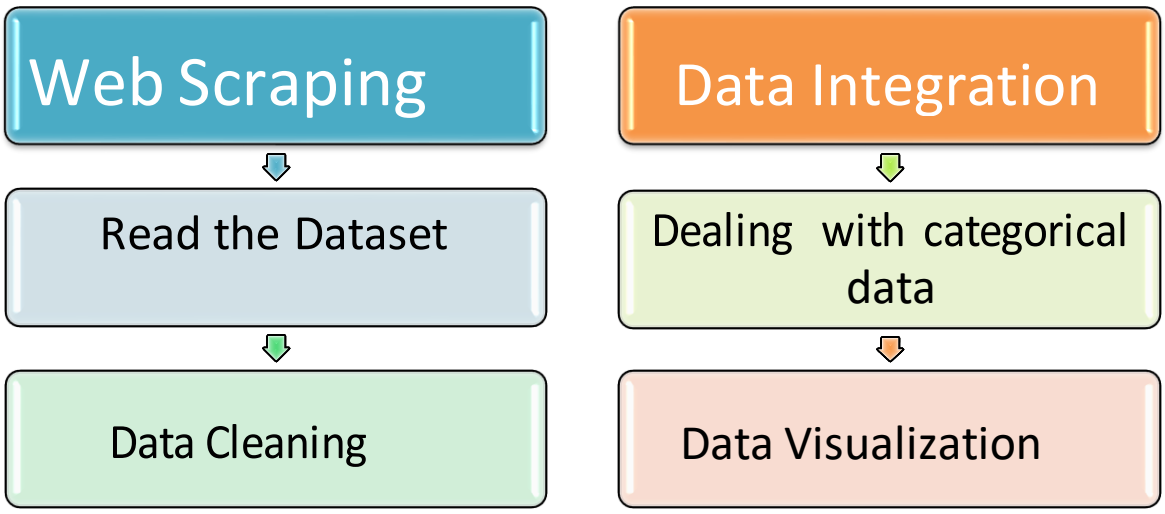
Project Outline :

Web scraping is an automatic method to obtain large amounts of data from websites. A spreadsheet or an API, for example. There are many techniques for web scraping. we have used r studio tool to scraping our data table. We used a IMDB websites for our Scrapping after that pre-processed the data.

Data pre-processing techniques are used when the data is inconsistent, which indicates that the data is not recorded in accordance with the restrictions on the column, noisy, which may contain a variety of mistakes or outliers, and incomplete, which indicates that some attribute value is missing. In our given dataset First,we tried to fix our missing values. After changing format, we have addeda new column using another column value what had been given in our question condition. After adding column then we handled the categorical value to numerical value for our discretization part. At last,we tried to visualize these data.

Data Visualization is the approach used to offer patterns in the data using visual cues such as graphs, charts, maps, and many more. we have used scatter plot to represent our data visualization.

Project Solution Design:



After creating new project in RStudio read the .CSV file and then done all the data pre-processing systems.

Web Scraping:

```
library(rvest)

ipl_players_df = data.frame()


csk_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/58/IPL'
rcb_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/59/IPL'
dc_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/61/IPL'
mi_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/62/IPL'
kkr_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/63/IPL'
rr_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/64/IPL'
pbks_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/65/IPL'
srh_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/255/IPL'
lsg_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/966/IPL'
gt_link = 'https://www.cricbuzz.com/api/html/series/5945/most-runs/3/971/IPL'


links = c(csk_link, rcb_link, dc_link, mi_link, kkr_link, rr_link, pbks_link, srh_link, lsg_link, gt_link)
teams = c('CSK', 'RCB', 'DC', 'MI', 'KKR', 'RR', 'PBKS', 'SRH', 'LSG', 'GT')

for (link in links) {
  page = read_html(link)
  table_body = html_nodes(page, 'tbody')

  table_rows = html_nodes(table_body, 'tr')

  player_name = html_nodes(table_rows, 'td:nth-child(2)')

  # player innings in 4th column
  player_innings = html_nodes(table_rows, 'td:nth-child(4)')

  # player runs in 5th column
  player_runs = html_nodes(table_rows, 'td:nth-child(5)')

  # player average in 6th column
  player_average = html_nodes(table_rows, 'td:nth-child(6)')

  # player strike rate in 7th column
  player_strike_rate = html_nodes(table_rows, 'td:nth-child(7)')

  # append the data to the data frame
  ipl_players_df = rbind(ipl_players_df, data.frame(
    team = teams[links == link],
    name = html_text(player_name),
    innings = html_text(player_innings),
    runs = html_text(player_runs),
    average = html_text(player_average),
    strike_rate = html_text(player_strike_rate)
  ))
}
```

This code is scrapes data from the most runs section of the IPL (Indian Premier League) website for each team and stores the data in a data frame.

It starts by creating a blank data frame called ipl_players_df. Then, it defines the links for each team's most runs section and stores them in a vector called links. It also defines the names of each team and stores them in a vector called teams.

Then, it starts a for loop that iterates through each link in the links vector. For each link, it reads the HTML content using the read_html() function from the rvest package. It then extracts the table body using the html_nodes() function and the 'tbody' selector. From the table body, it extracts the table rows using the html_nodes() function and the 'tr' selector.

For each row, it extracts the player name, innings, runs, average, and strike rate using the html_nodes() function and the appropriate nth-child selector. It then appends this data to the ipl_players_df data frame using the rbind() function and a new data frame created using the data.frame() function.

The final output of this code is a data frame containing the most runs data for each player on each team in the IPL.

```
> ipl_players_df
    team                        name innings runs average strike_rate
1    CSK                 Devon Conway       9  414   59.14      144.25
2    CSK               Ruturaj Gaikwad       9  354   44.25      145.68
3    CSK                  Shivam Dube       8  264   33.00      158.08
4    CSK                Ajinkya Rahane       6  224   44.80      189.83
5    CSK                     Moeen Ali       7  107   21.40      146.58
6    CSK               Ravindra Jadeja       7   92   18.40      143.75
7    CSK                 Ambati Rayudu       7   83   16.60      136.07
8    CSK                     MS Dhoni       6   74   74.00      211.43
9    CSK                    Ben Stokes       2   15    7.50      107.14
10   CSK                Mitchell Santner       2    2       -       50.00
11   RCB                  Faf du Plessis       9  466   58.25      159.59
12   RCB                     Virat Kohli       9  364   45.50      137.88
13   RCB                   Glenn Maxwell       9  262   32.75      183.22
14   RCB                  Dinesh Karthik       9   99   12.38      133.78
15   RCB                  Mahipal Lomror       6   78   15.60      130.00
16   RCB                   Shahbaz Ahmed       6   42   10.50      107.69
17   RCB                    David Willey       3   35       -      109.38
18   RCB              Suyash Prabhudessai       4   35    8.75      120.69
19   RCB                     Anuj Rawat       3   25   12.50       65.79
20   RCB                Wanindu Hasaranga       4   21   10.50      105.00
21   RCB                Michael Bracewell       1   19   19.00      105.56
22   RCB                     Akash Deep       1   17   17.00      212.50
23   RCB                Vijaykumar Vyshak       2   13   13.00      144.44
24   RCB                    Harshal Patel       2    6    3.00      100.00
25   RCB                     Karn Sharma       2    3    1.50       60.00
26    DC                   David Warner       9  308   34.22      118.46
27    DC                     Axar Patel       9  238   34.00      133.71
28    DC                   Manish Pandey       7  133   19.00      116.67
29    DC                  Mitchell Marsh       6   94   15.67      130.56
30    DC                 Aman Hakim Khan       6   85   14.17      119.72
31    DC                     Philip Salt       4   64   16.00      160.00
32    DC                   Sarfaraz Khan       4   53   13.25       85.48
33    DC                   Rilee Rossouw       4   52   13.00      133.33
34    DC                     Lalit Yadav       4   48   16.00      114.29
35    DC                    Prithvi Shaw       6   47    7.83      117.50
36    DC                     Ripal Patel       3   39   19.50      144.44
37    DC                   Anrich Nortje       6   37   12.33      132.14
38    DC                   Abishek Porel       4   33    8.25      106.45
39    DC                     Priyam Garg       2   22   11.00       95.65
40    DC                   Kuldeep Yadav       7   21   21.00       72.41
41    MI                     Tilak Varma       8  248   41.33      152.15
42    MI                   Cameron Green       8  243   48.60      152.83
43    MI                    Ishan Kishan       8  211   26.38      128.66
44    MI                 Suryakumar Yadav       8  201   25.12      176.32
45    MI                    Rohit Sharma       8  184   23.00      132.37
46    MI                       Tim David       8  158   39.50      169.89
47    MI                   Nehal Wadhera       4   67   16.75      171.79
48    MI                   Piyush Chawla       2   23   23.00      127.78
49    MI                  Hrithik Shokeen       2   23   23.00      143.75
50    MI                     Arshad Khan       2   17   17.00      130.77
51    MI                  Arjun Tendulkar       1   13   13.00      144.44
```

52	MI	Tristan Stubbs	1	5	5.00	50.00
53	MI	Jason Behrendorff	1	3	-	75.00
54	MI	Jofra Archer	1	1	-	50.00
55	KKR	Venkatesh Iyer	9	296	32.89	148.74
56	KKR	Rinku Singh	9	270	54.00	151.69
57	KKR	Nitish Rana	9	233	25.89	152.29
58	KKR	Rahmanullah Gurbaz	6	183	30.50	145.24
59	KKR	Jason Roy	3	160	53.33	170.21
60	KKR	Andre Russell	9	142	20.29	146.39
61	KKR	Shardul Thakur	6	101	20.20	183.64
62	KKR	N Jagadeesan	6	89	14.83	109.88
63	KKR	David Wiese	3	21	21.00	190.91
64	KKR	Umesh Yadav	5	19	9.50	105.56
65	KKR	Mandeep Singh	3	14	4.67	87.50
66	KKR	Sunil Narine	7	13	3.25	81.25
67	KKR	Litton Das	1	4	4.00	100.00
68	KKR	Anukul Roy	2	4	2.00	66.67
69	KKR	Varun Chakaravarthy	2	1	1.00	11.11
70	RR	Yashasvi Jaiswal	9	428	47.56	159.70
71	RR	Jos Buttler	9	289	32.11	138.94
72	RR	Sanju Samson	9	212	23.56	150.35
73	RR	Shimron Hetmyer	9	204	40.80	154.55
74	RR	Devdutt Padikkal	8	194	27.71	125.97
75	RR	Dhruv Jurel	8	132	26.40	191.30
76	RR	Ravichandran Ashwin	8	65	16.25	144.44
77	RR	Riyan Parag	5	54	13.50	112.50
78	RR	Jason Holder	3	12	6.00	109.09
79	RR	Abdul Basith	1	1	-	100.00
80	RR	Adam Zampa	1	1	1.00	100.00
81	PBKS	Shikhar Dhawan	6	262	65.50	148.86
82	PBKS	Prabhsimran Singh	9	210	23.33	153.28
83	PBKS	Sam Curran	9	192	27.43	143.28
84	PBKS	Jitesh Sharma	9	190	21.11	162.39
85	PBKS	Sikandar Raza	6	128	25.60	140.66
86	PBKS	Atharva Taide	5	112	22.40	149.33
87	PBKS	Matthew Short	5	90	18.00	136.36
88	PBKS	Shahrukh Khan	9	86	17.20	153.57
89	PBKS	Harpreet Singh Bhatia	3	76	25.33	128.81
90	PBKS	Liam Livingstone	4	75	18.75	138.89
91	PBKS	Bhanuka Rajapaksa	3	71	35.50	120.34
92	PBKS	Harpreet Brar	5	33	8.25	126.92
93	PBKS	Arshdeep Singh	2	2	-	100.00
94	PBKS	Nathan Ellis	2	1	0.50	14.29
95	PBKS	Mohit Rathee	1	1	-	50.00
96	SRH	Rahul Tripathi	8	170	24.29	114.86
97	SRH	Mayank Agarwal	8	169	21.12	110.46
98	SRH	Harry Brook	8	163	23.29	125.38
99	SRH	Heinrich Klaasen	5	153	51.00	182.14
100	SRH	Abhishek Sharma	6	139	23.17	156.18
101	SRH	Aiden Markram	7	132	22.00	138.95
102	SRH	Abdul Samad	4	90	45.00	120.00
103	SRH	Washington Sundar	5	60	15.00	100.00
104	SRH	Marco Jansen	3	32	32.00	103.23
105	SRH	Anmolpreet Singh	1	31	31.00	119.23
106	SRH	Adil Rashid	2	22	11.00	137.50
107	SRH	Umran Malik	2	19	19.00	237.50
108	SRH	Akeal Hosein	1	16	-	160.00
109	SRH	Glenn Phillips	1	8	8.00	133.33
110	SRH	Bhuvneshwar Kumar	3	8	4.00	53.33
111	LSG	Kyle Mayers	9	297	33.00	158.82
112	LSG	KL Rahul	9	274	34.25	113.22
113	LSG	Marcus Stoinis	9	229	28.62	144.94
114	LSG	Nicholas Pooran	9	225	28.12	190.68
115	LSG	Ayush Badoni	8	132	18.86	132.00
116	LSG	Krunal Pandya	9	122	20.33	119.61
117	LSG	Deepak Hooda	9	53	6.62	89.83
118	LSG	Krishnappa Gowtham	4	47	23.50	174.07
119	LSG	Amit Mishra	1	19	19.00	63.33
120	LSG	Naveen-ul-Haq	1	13	13.00	100.00
121	LSG	Mark Wood	2	11	11.00	220.00
122	LSG	Ravi Bishnoi	4	11	11.00	78.57
123	LSG	Jaydev Unadkat	1	9	9.00	128.57
124	LSG	Yudhvir Singh Charak	2	1	0.50	50.00
125	GT	Shubman Gill	9	339	37.67	140.66
126	GT	Hardik Pandya	8	213	30.43	117.03
127	GT	Vijay Shankar	7	205	41.00	158.91
128	GT	David Miller	8	180	45.00	147.54
129	GT	Sai Sudharsan	5	176	44.00	123.94
130	GT	Wriddhiman Saha	9	151	16.78	123.77
131	GT	Abhinav Manohar	5	112	22.40	140.00
132	GT	Rahul Tewatia	6	63	63.00	203.23
133	GT	Rashid Khan	4	16	16.00	228.57

Data Integration:

```
#Data Integration
# Define the new data
new_data <- data.frame(
  team = c("CSK", "MI", "MI"),
  name = c("Mitchell Santner", "Behrendorff Jason Behrendorff", "Jofra Archer"),
  innings = c(2, 1, 1),
  runs = c(1, 3, 1),
  average = c(2.00, 3.00, 1.00),
  strike_rate = c(50.00, 75.00, 50.00)
)

# Bind the new data to the existing data frame
ipl_players_df <- rbind(ipl_players_df, new_data)
```

The code is performing data integration, which involves combining new data with an existing data frame called ipl_players_df.

The first step is to define the new data using the data.frame function.
The next step is to bind the new data to the existing data frame ipl_players_df. This is done using the rbind function, which stands for "row bind". The rbind function is used to concatenate rows of two or more data frames with the same column names. In this case, the new_data data frame is appended to the bottom of the ipl_players_df data frame. This results in a new ipl_players_df data frame with additional rows representing the performance of the three players in their respective matches.

Finally, the updated ipl_players_df data frame is assigned to the same variable name ipl_players_df to reflect the changes.

```
> ipl_players_df
```

	team	name	innings	runs	average	strike_rate
1	CSK	Devon Conway	9	414	59.14	144.25
2	CSK	Ruturaj Gaikwad	9	354	44.25	145.68
3	CSK	Shivam Dube	8	264	33.00	158.08
4	CSK	Ajinkya Rahane	6	224	44.80	189.83
5	CSK	Moeen Ali	7	107	21.40	146.58
6	CSK	Ravindra Jadeja	7	92	18.40	143.75
7	CSK	Ambati Rayudu	7	83	16.60	136.07
8	CSK	MS Dhoni	6	74	74.00	211.43
9	CSK	Ben Stokes	2	15	7.50	107.14
10	CSK	Mitchell Santner	2	2	-	50.00
11	RCB	Faf du Plessis	9	466	58.25	159.59
12	RCB	Virat Kohli	9	364	45.50	137.88
13	RCB	Glenn Maxwell	9	262	32.75	183.22
14	RCB	Dinesh Karthik	9	99	12.38	133.78
15	RCB	Mahipal Lomror	6	78	15.60	130.00
16	RCB	Shahbaz Ahmed	6	42	10.50	107.69
17	RCB	David Willey	3	35	-	109.38
18	RCB	Suyash Prabhudessai	4	35	8.75	120.69
19	RCB	Anuj Rawat	3	25	12.50	65.79
20	RCB	Wanindu Hasaranga	4	21	10.50	105.00
21	RCB	Michael Bracewell	1	19	19.00	105.56
22	RCB	Akash Deep	1	17	17.00	212.50
23	RCB	Vijaykumar Vyshak	2	13	13.00	144.44
24	RCB	Harshal Patel	2	6	3.00	100.00
25	RCB	Karn Sharma	2	3	1.50	60.00
26	DC	David Warner	9	308	34.22	118.46
27	DC	Axar Patel	9	238	34.00	133.71
28	DC	Manish Pandey	7	133	19.00	116.67
29	DC	Mitchell Marsh	6	94	15.67	130.56
30	DC	Aman Hakim Khan	6	85	14.17	119.72
31	DC	Philip Salt	4	64	16.00	160.00
32	DC	Sarfaraz Khan	4	53	13.25	85.48
33	DC	Rilee Rossouw	4	52	13.00	133.33
34	DC	Lalit Yadav	4	48	16.00	114.29
35	DC	Prithvi Shaw	6	47	7.83	117.50
36	DC	Ripal Patel	3	39	19.50	144.44
37	DC	Anrich Nortje	6	37	12.33	132.14
38	DC	Abishek Porel	4	33	8.25	106.45
39	DC	Priyam Garg	2	22	11.00	95.65
40	DC	Kuldeep Yadav	7	21	21.00	72.41
41	MI	Tilak Varma	8	248	41.33	152.15
42	MI	Cameron Green	8	243	48.60	152.83
43	MI	Ishan Kishan	8	211	26.38	128.66
44	MI	Suryakumar Yadav	8	201	25.12	176.32
45	MI	Rohit Sharma	8	184	23.00	132.37
46	MI	Tim David	8	158	39.50	169.89
47	MI	Nehal Wadhera	4	67	16.75	171.79
48	MI	Piyush Chawla	2	23	23.00	127.78
49	MI	Hrithik Shokeen	2	23	23.00	143.75
50	MI	Arshad Khan	2	17	17.00	130.77
51	MI	Arjun Tendulkar	1	13	13.00	144.44

52	MI	Tristan Stubbs	1	5	5.00	50.00
53	MI	Jason Behrendorff	1	3	-	75.00
54	MI	Jofra Archer	1	1	-	50.00
55	KKR	Venkatesh Iyer	9	296	32.89	148.74
56	KKR	Rinku Singh	9	270	54.00	151.69
57	KKR	Nitish Rana	9	233	25.89	152.29
58	KKR	Rahmanullah Gurbaz	6	183	30.50	145.24
59	KKR	Jason Roy	3	160	53.33	170.21
60	KKR	Andre Russell	9	142	20.29	146.39
61	KKR	Shardul Thakur	6	101	20.20	183.64
62	KKR	N Jagadeesan	6	89	14.83	109.88
63	KKR	David Wiese	3	21	21.00	190.91
64	KKR	Umesh Yadav	5	19	9.50	105.56
65	KKR	Mandeep Singh	3	14	4.67	87.50
66	KKR	Sunil Narine	7	13	3.25	81.25
67	KKR	Litton Das	1	4	4.00	100.00
68	KKR	Anukul Roy	2	4	2.00	66.67
69	KKR	Varun Chakaravarthy	2	1	1.00	11.11
70	RR	Yashasvi Jaiswal	9	428	47.56	159.70
71	RR	Jos Buttler	9	289	32.11	138.94
72	RR	Sanju Samson	9	212	23.56	150.35
73	RR	Shimron Hetmyer	9	204	40.80	154.55
74	RR	Devdutt Padikkal	8	194	27.71	125.97
75	RR	Dhruv Jurel	8	132	26.40	191.30
76	RR	Ravichandran Ashwin	8	65	16.25	144.44
77	RR	Riyan Parag	5	54	13.50	112.50
78	RR	Jason Holder	3	12	6.00	109.09
79	RR	Abdul Basith	1	1	-	100.00
80	RR	Adam Zampa	1	1	1.00	100.00
81	PBKS	Shikhar Dhawan	6	262	65.50	148.86
82	PBKS	Prabhsimran Singh	9	210	23.33	153.28
83	PBKS	Sam Curran	9	192	27.43	143.28
84	PBKS	Jitesh Sharma	9	190	21.11	162.39
85	PBKS	Sikandar Raza	6	128	25.60	140.66
86	PBKS	Atharva Taide	5	112	22.40	149.33
87	PBKS	Matthew Short	5	90	18.00	136.36
88	PBKS	Shahrukh Khan	9	86	17.20	153.57
89	PBKS	Harpreet Singh Bhatia	3	76	25.33	128.81
90	PBKS	Liam Livingstone	4	75	18.75	138.89
91	PBKS	Bhanuka Rajapaksa	3	71	35.50	120.34
92	PBKS	Harpreet Brar	5	33	8.25	126.92
93	PBKS	Arshdeep Singh	2	2	-	100.00
94	PBKS	Nathan Ellis	2	1	0.50	14.29
95	PBKS	Mohit Rathee	1	1	-	50.00
96	SRH	Rahul Tripathi	8	170	24.29	114.86
97	SRH	Mayank Agarwal	8	169	21.12	110.46
98	SRH	Harry Brook	8	163	23.29	125.38
99	SRH	Heinrich Klaasen	5	153	51.00	182.14
100	SRH	Abhishek Sharma	6	139	23.17	156.18
101	SRH	Aiden Markram	7	132	22.00	138.95
102	SRH	Abdul Samad	4	90	45.00	120.00
103	SRH	Washington Sundar	5	60	15.00	100.00
104	SRH	Marco Jansen	3	32	32.00	103.23
105	SRH	Anmolpreet Singh	1	31	31.00	119.23
106	SRH	Adil Rashid	2	22	11.00	137.50
107	SRH	Umran Malik	2	19	19.00	237.50
108	SRH	Akeal Hosein	1	16	-	160.00
109	SRH	Glenn Phillips	1	8	8.00	133.33
110	SRH	Bhuvneshwar Kumar	3	8	4.00	53.33
111	LSG	Kyle Mayers	9	297	33.00	158.82
112	LSG	KL Rahul	9	274	34.25	113.22
113	LSG	Marcus Stoinis	9	229	28.62	144.94
114	LSG	Nicholas Pooran	9	225	28.12	190.68
115	LSG	Ayush Badoni	8	132	18.86	132.00
116	LSG	Krunal Pandya	9	122	20.33	119.61
117	LSG	Deepak Hooda	9	53	6.62	89.83
118	LSG	Krishnappa Gowtham	4	47	23.50	174.07
119	LSG	Amit Mishra	1	19	19.00	63.33
120	LSG	Naveen-ul-Haq	1	13	13.00	100.00
121	LSG	Mark Wood	2	11	11.00	220.00
122	LSG	Ravi Bishnoi	4	11	11.00	78.57
123	LSG	Jaydev Unadkat	1	9	9.00	128.57
124	LSG	Yudhvir Singh Charak	2	1	0.50	50.00
125	GT	Shubman Gill	9	339	37.67	140.66
126	GT	Hardik Pandya	8	213	30.43	117.03
127	GT	Vijay Shankar	7	205	41.00	158.91
128	GT	David Miller	8	180	45.00	147.54
129	GT	Sai Sudharsan	5	176	44.00	123.94
130	GT	Wriddhiman Saha	9	151	16.78	123.77
131	GT	Abhinav Manohar	5	112	22.40	140.00
132	GT	Rahul Tewatia	6	63	63.00	203.23
133	GT	Rashid Khan	4	16	16.00	228.57
134	CSK	Mitchell Santner	2	1	2	50
135	MI	Behrendorff Jason Behrendorff	1	3	3	75
136	MI	Jofra Archer	1	1	1	50

```
>
```

preprocessing the data:

Data Cleaning:

- **Smooth noisy data**

```
### Smooth noisy data

ipl_players_df$runs = as.numeric(gsub('[-]', NA, ipl_players_df$runs))
ipl_players_df$average = as.numeric(gsub('[-]', NA, ipl_players_df$average))
ipl_players_df$strike_rate = as.numeric(gsub('[-]', NA, ipl_players_df$strike_rate))

### Remove duplicates

ipl_players_df = ipl_players_df[!duplicated(ipl_players_df), ]

### Remove rows with NA values

ipl_players_df = na.omit(ipl_players_df)


## Remove rows with 0 runs

ipl_players_df = ipl_players_df[ipl_players_df$runs != 0, ]

## remove rows with 0 innings

ipl_players_df = ipl_players_df[ipl_players_df$innings != 0, ]

## remove rows with 0 average

ipl_players_df = ipl_players_df[ipl_players_df$average != 0, ]
```

These lines of code are cleaning the data by removing duplicates and rows with missing or zero values.

First, the code replaces any dash (-) characters in the "runs", "average", and "strike_rate" columns with NA values, and then converts the columns to numeric data types. This is done to ensure that these columns contain only numeric values and can be used for further analysis.

Next, the code removes duplicate rows from the data frame using the duplicated function. This function returns a logical vector indicating which rows are duplicated, and the ! operator negates the vector to select the non-duplicated rows.

Then, the code removes any rows with missing values using the na.omit function. This function removes any rows that contain missing values in any of the columns.

The code then removes any rows where the "runs" column has a value of 0, as these are likely to be incomplete or erroneous data. Similarly, it removes any rows where the "innings" or "average" columns have a value of 0, as these indicate that the player did not bat in any innings or did not score any runs on average. These rows are unlikely to be informative for analysis and can be safely removed.

- **Handle missing data**

```
## Handle missing data

### Replace missing values with mean

ipl_players_df$runs[is.na(ipl_players_df$runs)] = mean(ipl_players_df$runs, na.rm = TRUE)
ipl_players_df$average[is.na(ipl_players_df$average)] = mean(ipl_players_df$average, na.rm = TRUE)
ipl_players_df$strike_rate[is.na(ipl_players_df$strike_rate)] = mean(ipl_players_df$strike_rate, na.rm = TRUE)
```

This code is handling missing data in the ipl_players_df data frame.

In the first step, the code replaces the - characters in the runs, average, and strike_rate columns with NA values, and then converts those columns to numeric.

Next, the code removes any duplicate rows and any rows with NA values.

Then, the code removes any rows where the runs column is equal to 0, the innings column is equal to 0, or the average column is equal to 0.

Finally, the code handles any remaining missing values in the runs, average, and strike_rate columns by replacing them with the mean of the non-missing values in the respective columns, using the mean() function with the na.rm = TRUE argument to exclude any NA values from the calculation.

- **Data Munging**

```
# Data Munging

## Create a new column for player type

ipl_players_df$type = NA
```

This code adds a new column named "type" to the IPL players data frame and initializes it with NA values.

```
## Assign player type based on average and strike rate

ipl_players_df$type[(ipl_players_df$average > 10) & (ipl_players_df$strike_rate > 100)] = 'Batsman'
ipl_players_df$type[(ipl_players_df$runs < 10) & (ipl_players_df$strike_rate < 100)] = 'Bowler'
ipl_players_df$type[(ipl_players_df$average > 35) & (ipl_players_df$strike_rate > 140)] = 'Power Hitter'
ipl_players_df$type[(ipl_players_df$average >= 25) & (ipl_players_df$average <= 35) & (ipl_players_df$strike_rate > 130)] = 'Aggressive Batsman'
ipl_players_df$type[(ipl_players_df$average >= 25) & (ipl_players_df$average <= 35) & (ipl_players_df$strike_rate >= 100) & (ipl_players_df$strike_rate <= 130)] = 'Batting All-rounder'
ipl_players_df$type[(ipl_players_df$average < 25) & (ipl_players_df$strike_rate > 130)] = 'Slogger'
ipl_players_df$type[(ipl_players_df$average < 25) & (ipl_players_df$strike_rate >= 100) & (ipl_players_df$strike_rate <= 130)] = 'Bowling All-rounder'
ipl_players_df$type[(ipl_players_df$average < 15) & (ipl_players_df$strike_rate < 100)] = 'Tailender'


ipl_players_df



### Convert runs, average and strike rate to numeric

ipl_players_df$runs = as.numeric(ipl_players_df$runs)
ipl_players_df$average = as.numeric(ipl_players_df$average)
ipl_players_df$strike_rate = as.numeric(ipl_players_df$strike_rate)
```

This code assigns a player type to each player in the ipl_players_df dataframe based on their average and strike rate. The types are: "Batsman", "Bowler", "Power Hitter", "Aggressive Batsman", "Batting All-rounder", "Slogger", "Bowling All-rounder", and "Tailender". The code then converts the "runs", "average", and "strike_rate" columns to numeric data type.

```
> ipl_players_df
    team                        name innings runs average strike_rate                type
1    CSK                 Devon Conway       9  414   59.14      144.25         Power Hitter
2    CSK               Ruturaj Gaikwad       9  354   44.25      145.68         Power Hitter
3    CSK                  Shivam Dube       8  264   33.00      158.08  Aggressive Batsman
4    CSK                Ajinkya Rahane       6  224   44.80      189.83         Power Hitter
5    CSK                     Moeen Ali       7  107   21.40      146.58              Slogger
```

Data Discretization:

```
# Data Discretization

## Create a new column for player performance

ipl_players_df$performance = 'Average'

## Assign player performance based on runs, average and strike rate


mean_runs = mean(ipl_players_df$runs)
mean_average = mean(ipl_players_df$average)
mean_strike_rate = mean(ipl_players_df$strike_rate)

ipl_players_df$performance[ipl_players_df$runs > mean_runs*1.2 & ipl_players_df$average < mean_average & ipl_players_df$strike_rate < mean_strike_rate] = 'Inconsistent'
ipl_players_df$performance[ipl_players_df$runs < mean_runs & ipl_players_df$average > mean_average*1.2 & ipl_players_df$strike_rate < mean_strike_rate] = 'Steady'
ipl_players_df$performance[ipl_players_df$runs < mean_runs & ipl_players_df$average < mean_average & ipl_players_df$strike_rate > mean_strike_rate*1.2] = 'Impactful'
ipl_players_df$performance[ipl_players_df$runs > mean_runs & ipl_players_df$average > mean_average & ipl_players_df$strike_rate > mean_strike_rate] = 'Excellent'
ipl_players_df$performance[ipl_players_df$runs > mean_runs & ipl_players_df$average > mean_average & ipl_players_df$strike_rate < mean_strike_rate] = 'Good'
ipl_players_df$performance[ipl_players_df$runs > mean_runs & ipl_players_df$average < mean_average & ipl_players_df$strike_rate < mean_strike_rate] = 'Average'
ipl_players_df$performance[ipl_players_df$runs < mean_runs & ipl_players_df$average < mean_average & ipl_players_df$strike_rate > mean_strike_rate*0.8] = 'Poor'
ipl_players_df$performance[ipl_players_df$runs < mean_runs & ipl_players_df$average < mean_average & ipl_players_df$strike_rate < mean_strike_rate*0.8] = 'Bad'
ipl_players_df$performance[ipl_players_df$runs > mean_runs*1.2 & ipl_players_df$average > mean_average*1.2 & ipl_players_df$strike_rate > mean_strike_rate*1.2] = 'Dominant'
ipl_players_df$performance[ipl_players_df$runs > mean_runs*1.2 & ipl_players_df$average > mean_average*1.2 & ipl_players_df$strike_rate < mean_strike_rate] = 'Solid'
ipl_players_df$performance[ipl_players_df$runs < mean_runs*0.8 & ipl_players_df$average > mean_average & ipl_players_df$strike_rate > mean_strike_rate] = 'Consistent'
```

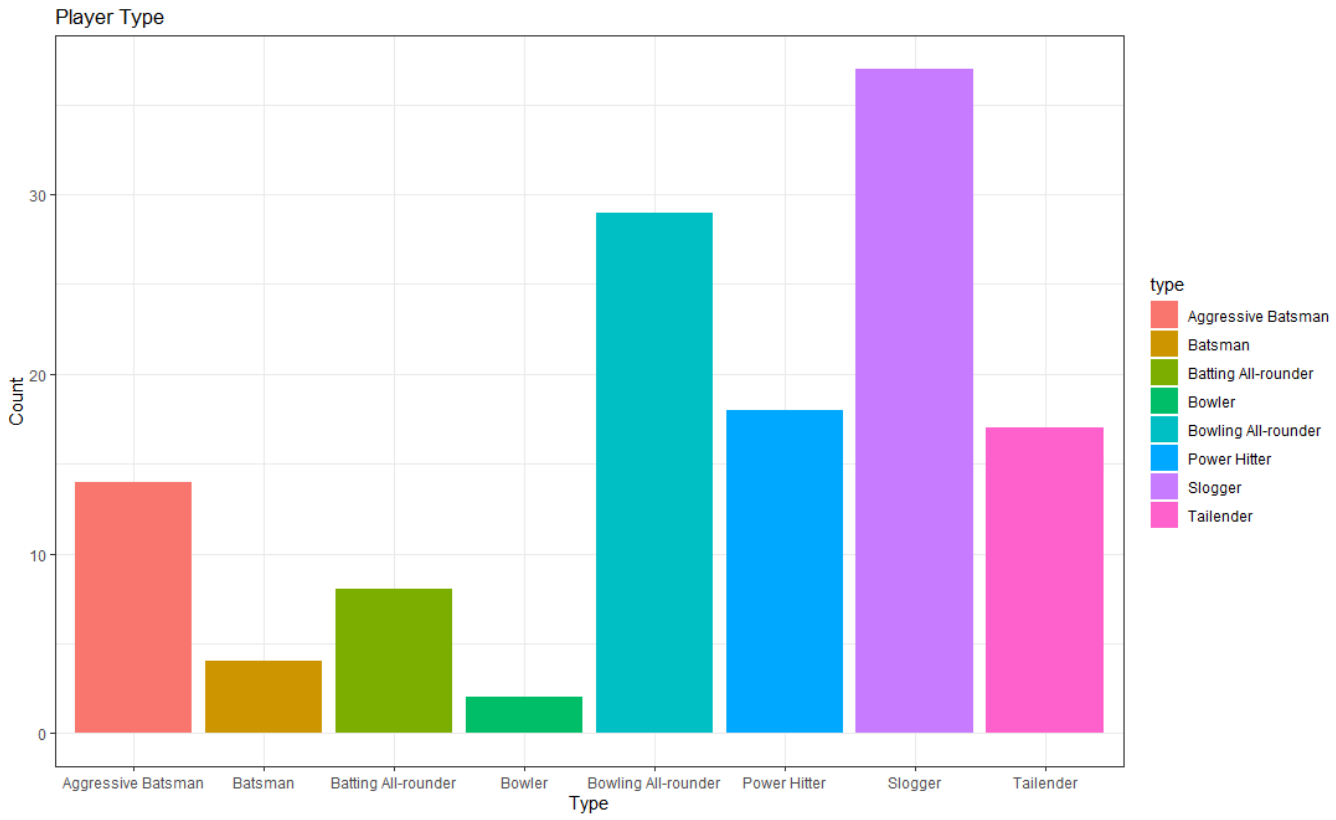
This code creates a new column called "performance" in the dataframe "ipl_players_df" and assigns the value "Average" to all rows in that column. Then, it assigns a different performance level to each row based on the player's runs, average, and strike rate compared to the mean values of those statistics in the entire dataframe. For example, if a player's runs are greater than 1.2 times the mean runs and their average and strike rate are less than the mean average and mean strike rate, respectively, then their performance level is "Inconsistent". Similarly, there are different conditions for assigning other performance levels such as "Steady", "Impactful", "Excellent", "Good", "Poor", "Bad", "Dominant", "Solid", and "Consistent".

```
> ipl_players_df
    team                 name innings runs average strike_rate               type performance
1    CSK          Devon Conway       9  414   59.14      144.25       Power Hitter   Excellent
2    CSK       Ruturaj Gaikwad       9  354   44.25      145.68       Power Hitter   Excellent
3    CSK           Shivam Dube       8  264   33.00      158.08 Aggressive Batsman    Dominant
4    CSK        Ajinkya Rahane       6  224   44.80      189.83       Power Hitter    Dominant
5    CSK             Moeen Ali       7  107   21.40      146.58             Slogger        Poor


128   GT          David Miller       8  180   45.00      147.54       Power Hitter   Excellent
129   GT          Sai Sudharsan      5  176   44.00      123.94            Batsman       Solid
130   GT       Wriddhiman Saha       9  151   16.78      123.77 Bowling All-rounder     Average
131   GT       Abhinav Manohar       5  112   22.40      140.00             Slogger     Average
132   GT         Rahul Tewatia       6   63   63.00      203.23       Power Hitter  Consistent
```

Data Visualization:

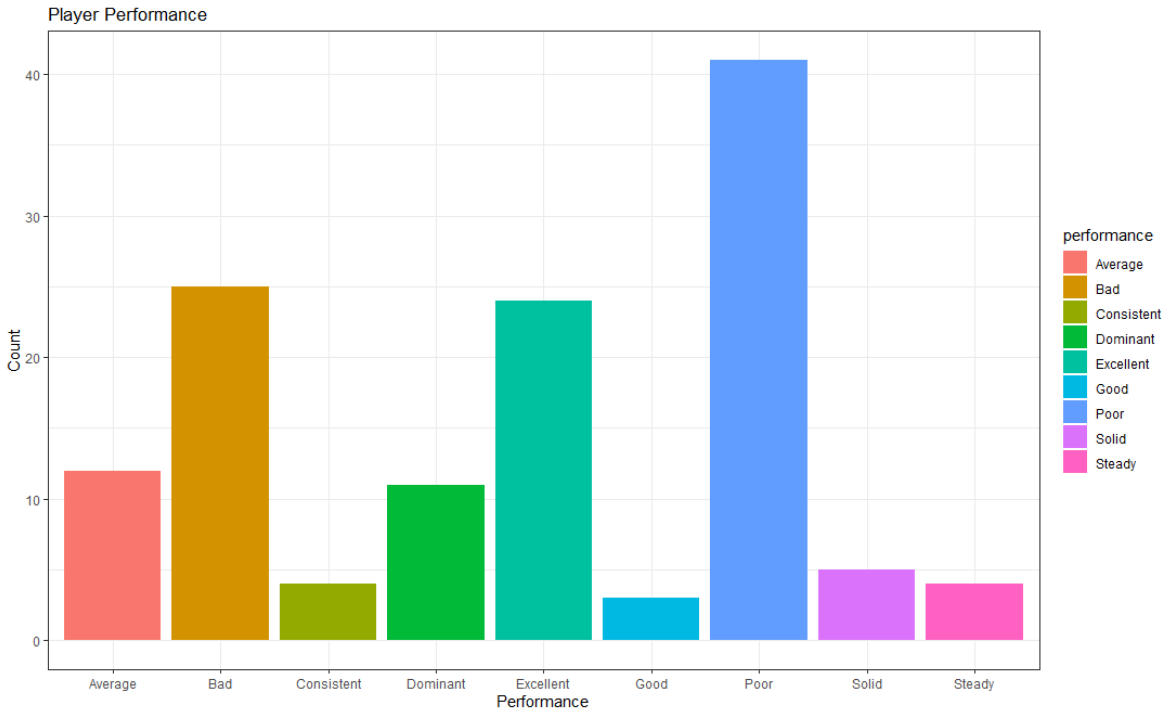
The below Bar plot for player type

```
## Bar plot for player type
ggplot(ipl_players_df, aes(x = type, fill = type)) + geom_bar() + theme_bw() + labs(title = 'Player Type', x = 'Type', y = 'Count')
```



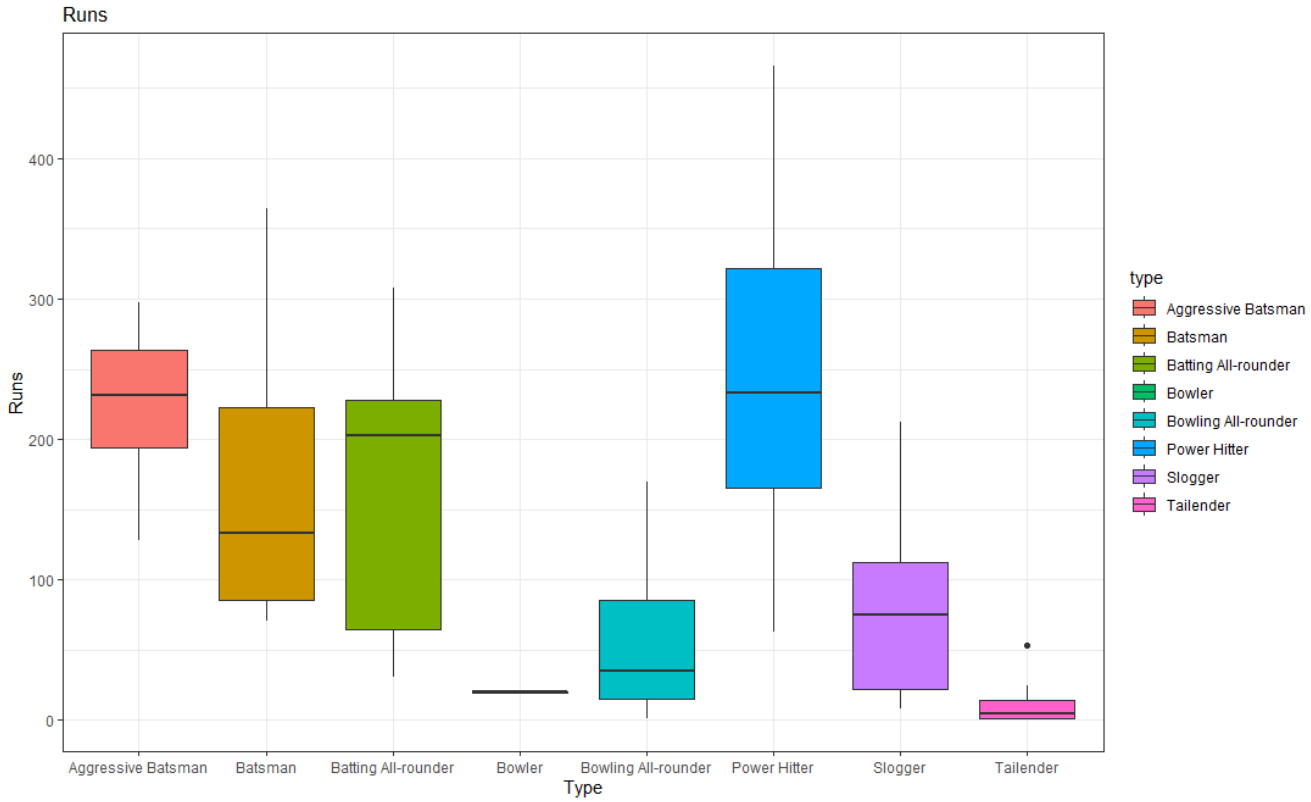
The below Bar plot for player type

```
## Bar plot for player performance
ggplot(ipl_players_df, aes(x = performance, fill = performance)) + geom_bar() + theme_bw() + labs(title = 'Player Performance', x = 'Performance', y = 'Count')
```



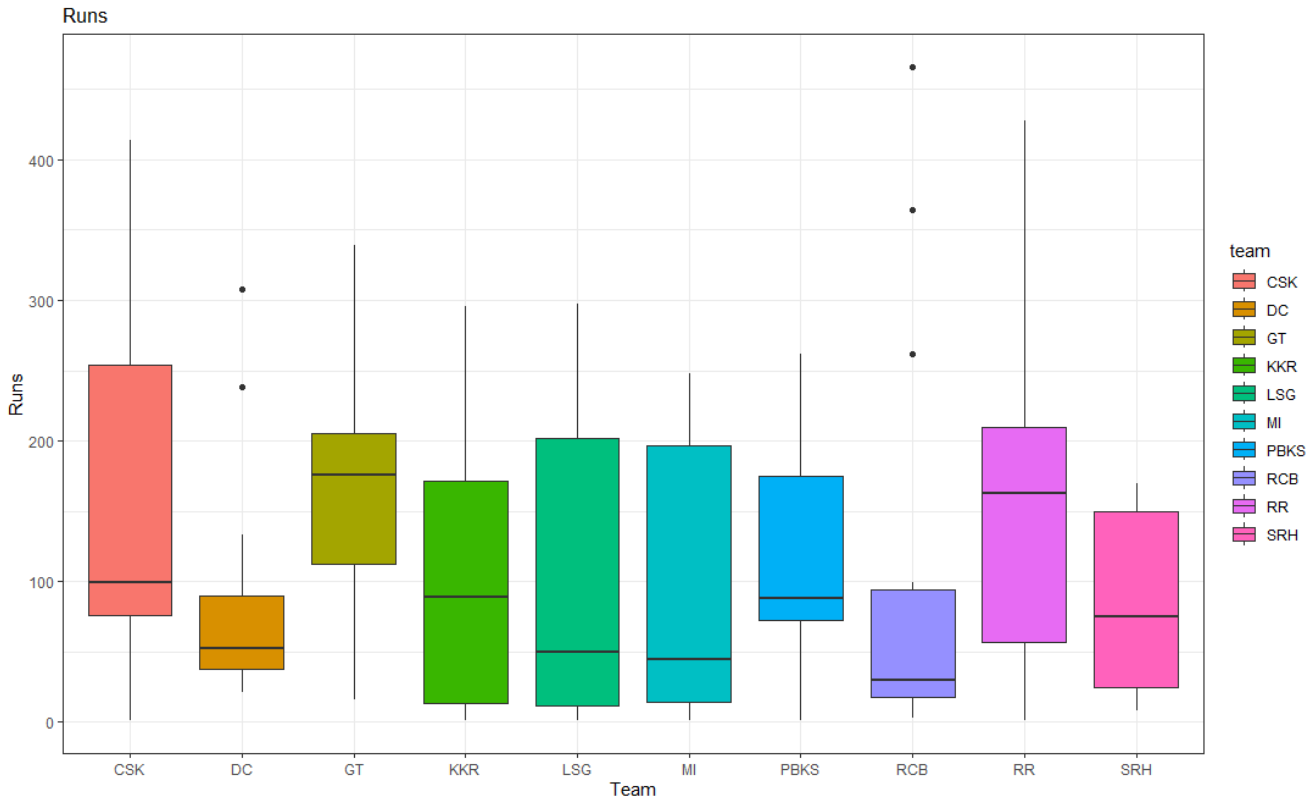
The below Box plot is made depending on the type(X-axis) and runs(Y-Axis)

```
## Box plot for runs
ggplot(ipl_players_df, aes(x = type, y = runs, fill = type)) + geom_boxplot() + theme_bw() + labs(title = 'Runs', x = 'Type', y = 'Runs')
```



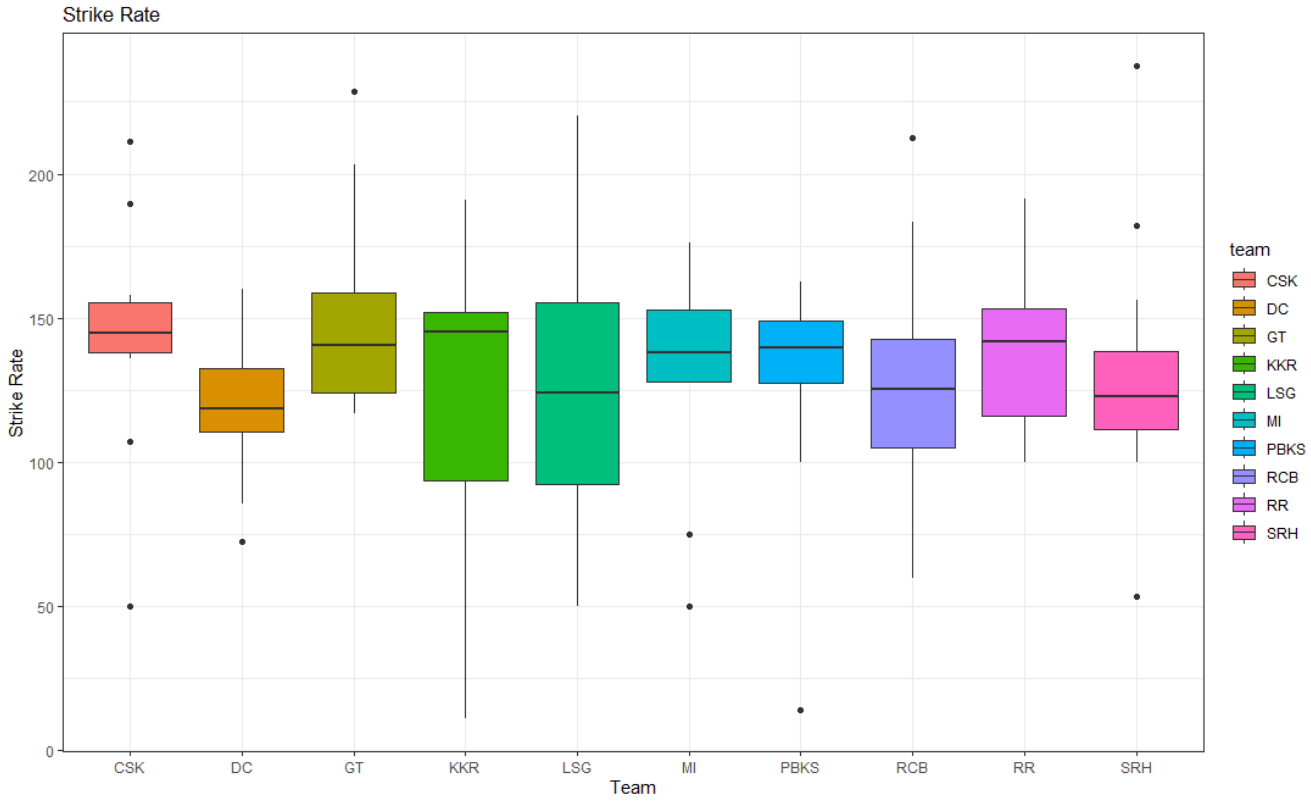
Measure the team performance by runs, strike rate, average, moving average and moving average of strike rate
The below Box plot is made depending on the team(X-axis) and runs(Y-Axis)

```
## Measure the team performance by runs, strike rate, average, moving average and moving average of strike rate

### Runs

ggplot(ipl_players_df, aes(x = team, y = runs, fill = team)) + geom_boxplot() + theme_bw() + labs(title = 'Runs', x = 'Team', y = 'Runs')
```



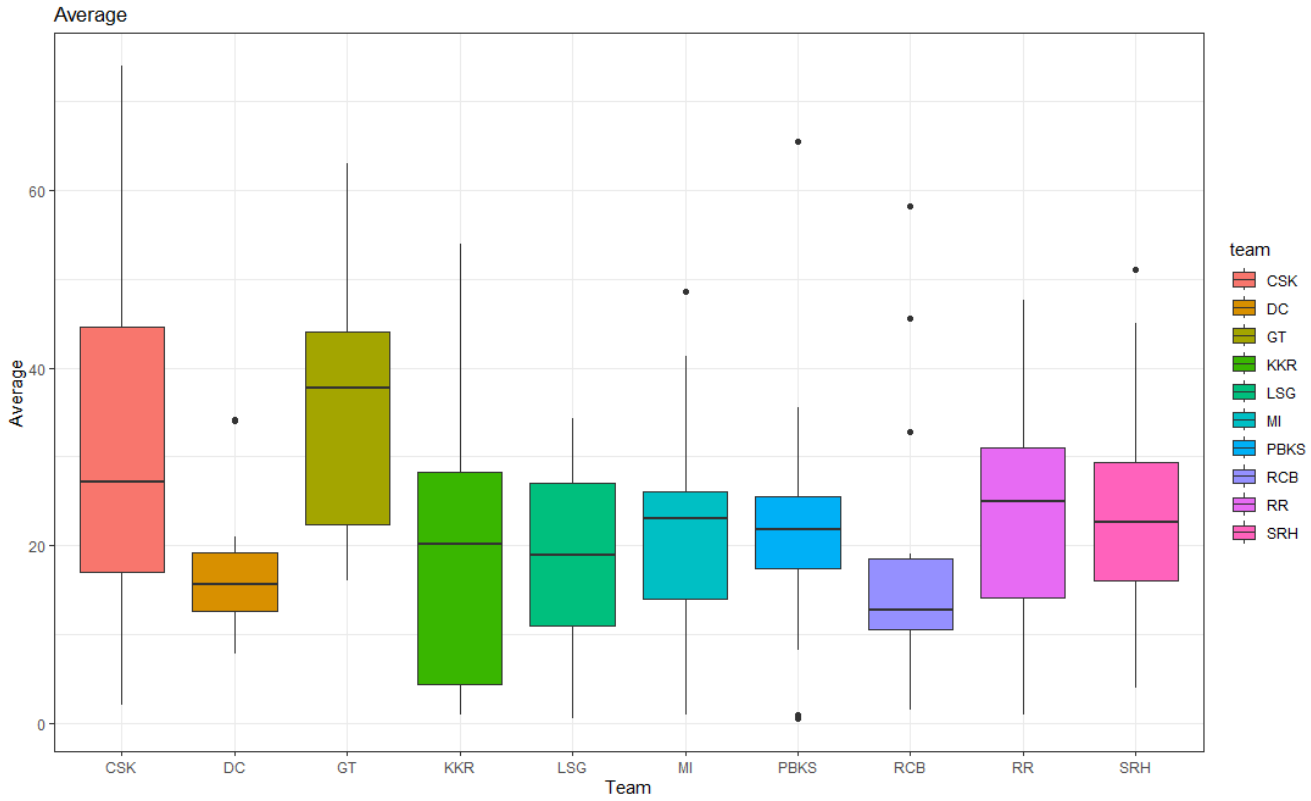
The below Box plot is made depending on the team(X-axis) and strike_rate(Y-Axis)

```
### Strike Rate

ggplot(ipl_players_df, aes(x = team, y = strike_rate, fill = team)) + geom_boxplot() + theme_bw() + labs(title = 'Strike Rate', x = 'Team', y = 'Strike Rate')
```



The below Box plot is made depending on the team(X-axis) and average(Y-Axis)

```
### Average

ggplot(ipl_players_df, aes(x = team, y = average, fill = team)) + geom_boxplot() + theme_bw() + labs(title = 'Average', x = 'Team', y = 'Average')


#
```


Average
60
40
20
0
Average
CSK
DC
GT
KKR
LSG
MI
PBKS
RCB
RR
SRH
Team
team
CSK
DC
GT
KKR
LSG
MI
PBKS
RCB
RR
SRH

Discussion and Conclusion:

The project plays a vital role to learn how to web scrape and pre-process data. We learned about webscraping, cleaning,integration,reduction,transformation and discretization of data from a datset.I faced some kind of difficulties working with the datas using RStudio but It gives us a few knowledge how a data scientists work with datas.Now a days,every information is a data and we are surrounding by datas.so,its important to know how data can be scraped and processed.