VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A1b: Preliminary preparation and analysis of data- Descriptive statistics

JYOTHIS KANIYAMPARAMBIL THANKACHAN

V01110144

Date of Submission: 18-06-2024

CONTENTS

INTRODUCTION

This study focuses to the performance of Cricket players who played in IPL from 2007/2008 to 2024, for this we are used IPL_ball_by_ball_updated till 2024 dataset and IPL SALARIES 2024 dataset to identify the relationship between a player's performance and the salary he gets in your data and Determine the best possible allocation of wickets taken and runs scored by the top three bowlers and batsmen in the three IPL tournaments that were lost also Sort the IPL statistics according to rounds and batter, ball, runs, and wickets for each player in each match. List the top three scorers in each IPL round, along with the top three wicket-takers. R is a potent statistical programming language well-known for its adaptability in managing and analysing big datasets, has been used to import the dataset.

By leveraging the strengths of Python and R, this analysis ensures a robust approach to unravelling the complexities of IPL player performance and their financial implications. This dual-framework approach not only facilitates rigorous statistical modelling but also enhances the interpretability of findings, providing a holistic view of the relationships between player performance and salary in the IPL context. The outcome of this analysis which help to easy under standing of correlation between player salary and runs.

OBJECTIVES

- Extract the files in R/Python
- Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and tow three wicket-takers in each IPL round.
- Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.
- Find the relationship between a player's performance and the salary he gets in your data.

BUSINESS SIGNIFICANCE

For IPL clubs, knowing the relationship between player performance and pay has significant business implications. Teams may create player contracts that optimise effort during games and guarantee that investments positively correspond with on-field success by implementing performance-based incentives. It also aids teams in more efficient budget allocation by highlighting the essential players that bring the greatest value. It also aids in talent scouting by promoting data-driven decisions in recruiting new talent based on statistical evidence of success. Franchises are able to more accurately estimate individual performances by fitting appropriate distributions for wickets and runs, which enables them to make better decisions on squad composition and strategic planning.

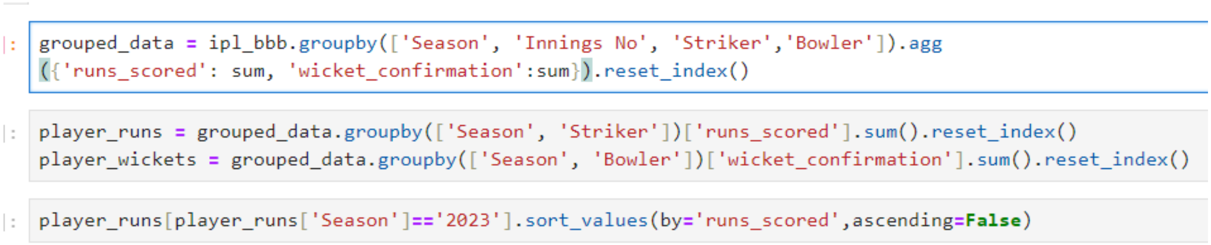
Results

- **Extract the files in R/Python**

Files containing code written in R and Python have been pushed to GitHub. This allows for efficient version control, collaboration, and sharing. Hosting the code on GitHub ensures that it is accessible to public.

- **Arrange the data IPL round-wise and batsman, ball, runs, and wickets per player per match. Indicate the top three run-getters and top three wicket-takers in each IPL round.**

Codes:

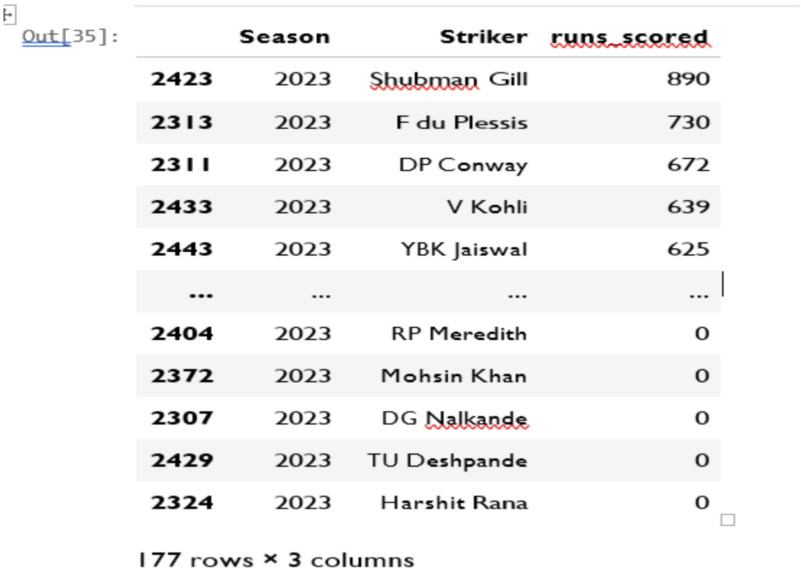


```
grouped_data = ipl_bbb.groupby(['Season', 'Innings No', 'Striker','Bowler']).agg
({'runs_scored': sum, 'wicket_confirmation':sum}).reset_index()
```

```
player_runs = grouped_data.groupby(['Season', 'Striker'])['runs_scored'].sum().reset_index()
player_wickets = grouped_data.groupby(['Season', 'Bowler'])['wicket_confirmation'].sum().reset_index()
```

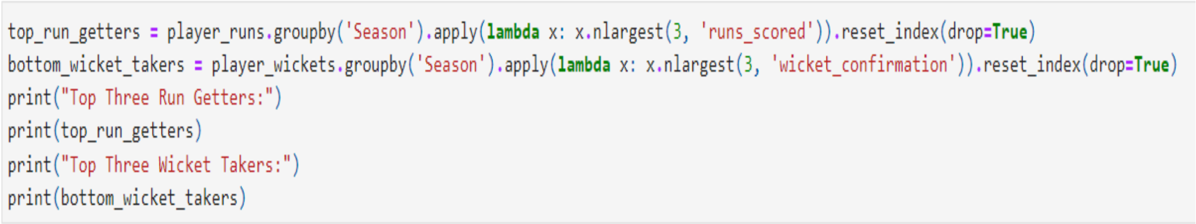
```
player_runs[player_runs['Season']=='2023'].sort_values(by='runs_scored',ascending=False)
```

Result:



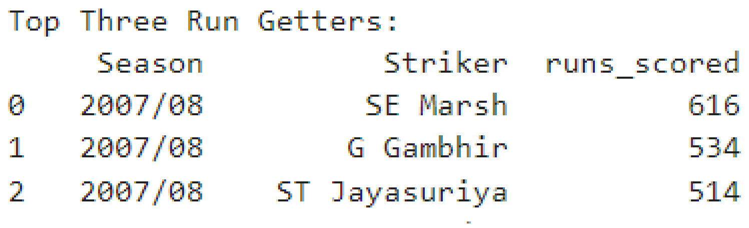
```
Out[35]:
        Season        Striker  runs_scored
2423      2023   Shubman Gill          890
2313      2023  F du Plessis           730
2311      2023      DP Conway          672
2433      2023        V Kohli          639
2443      2023    YBK Jaiswal          625
 ...       ...            ...          ...
2404      2023    RP Meredith            0
2372      2023    Mohsin Khan            0
2307      2023    DG Nalkande            0
2429      2023  TU Deshpande             0
2324      2023   Harshit Rana            0

177 rows × 3 columns
```

Codes of top three run-getters:

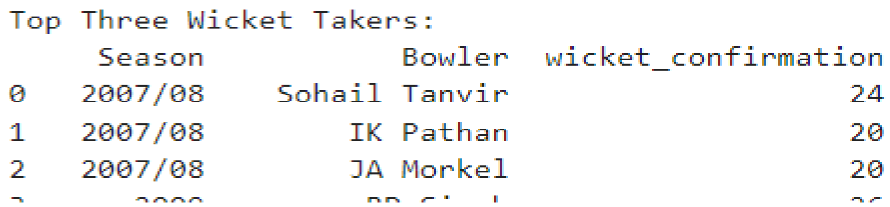


```
top_run_getters = player_runs.groupby('Season').apply(lambda x: x.nlargest(3, 'runs_scored')).reset_index(drop=True)
bottom_wicket_takers = player_wickets.groupby('Season').apply(lambda x: x.nlargest(3, 'wicket_confirmation')).reset_index(drop=True)
print("Top Three Run Getters:")
print(top_run_getters)
print("Top Three Wicket Takers:")
print(bottom_wicket_takers)
```

Results:



```
Top Three Run Getters:
    Season          Striker  runs_scored
0   2007/08         SE Marsh          616
1   2007/08       G Gambhir          534
2   2007/08    ST Jayasuriya          514
```



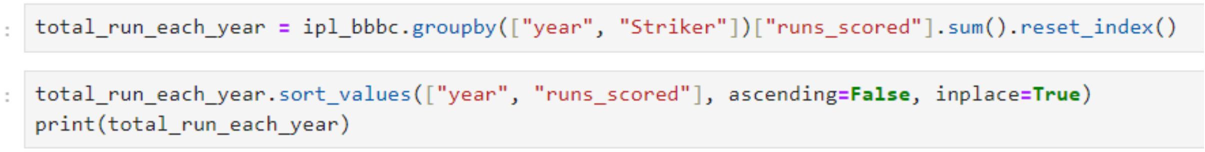
```
Top Three Wicket Takers:
    Season          Bowler  wicket_confirmation
0   2007/08    Sohail Tanvir                   24
1   2007/08        IK Pathan                   20
2   2007/08        JA Morkel                   20
```

Interpretation

The data you sent is the result of a query about the top run getters and wicket takers in the 2007/2008 IPL season, not rounds. It does not show batsman, ball, runs, and wickets per player per match.There is no data on batsman, balls bowled, runs conceded, or wickets taken per player per match.

- **Fit the most appropriate distribution for runs scored and wickets taken by the top three batsmen and bowlers in the lost three IPL tournaments.**

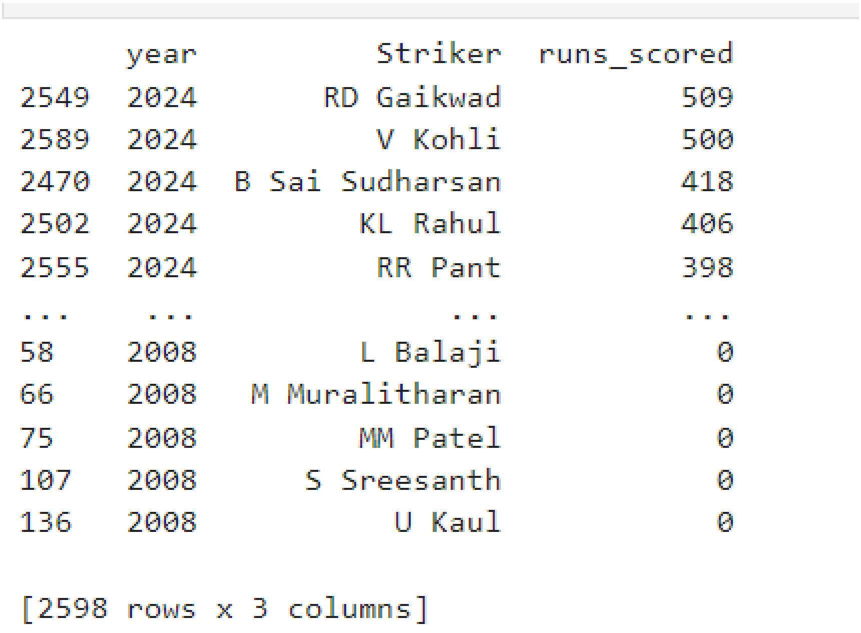
Code for runs scored



```
total_run_each_year = ipl_bbbc.groupby(["year", "Striker"])["runs_scored"].sum().reset_index()
```

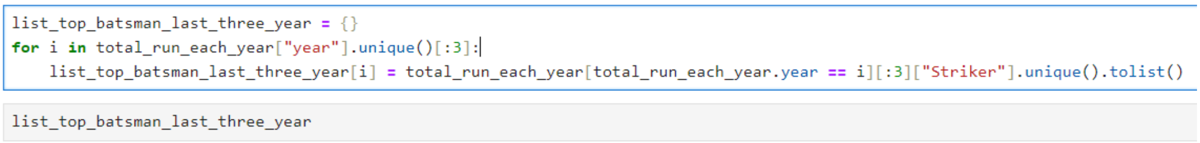
```
total_run_each_year.sort_values(["year", "runs_scored"], ascending=False, inplace=True)
print(total_run_each_year)
```

Result :



```
      year          Striker  runs_scored
2549  2024       RD Gaikwad          509
2589  2024          V Kohli          500
2470  2024  B Sai Sudharsan          418
2502  2024         KL Rahul          406
2555  2024          RR Pant          398
...    ...              ...          ...
58    2008         L Balaji            0
66    2008   M Muralitharan            0
75    2008         MM Patel            0
107   2008      S Sreesanth            0
136   2008           U Kaul            0

[2598 rows x 3 columns]
```

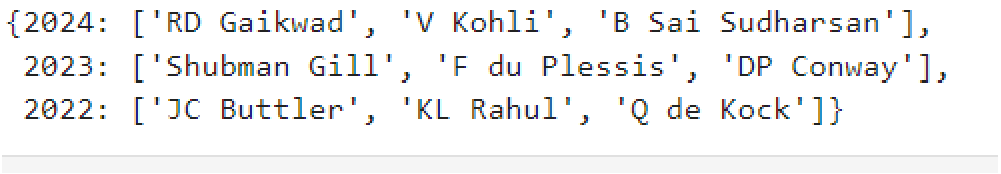
Code for top 3 runs scored:



```
list_top_batsman_last_three_year = {}
for i in total_run_each_year["year"].unique()[:3]:
    list_top_batsman_last_three_year[i] = total_run_each_year[total_run_each_year.year == i][:3]["Striker"].unique().tolist()
```

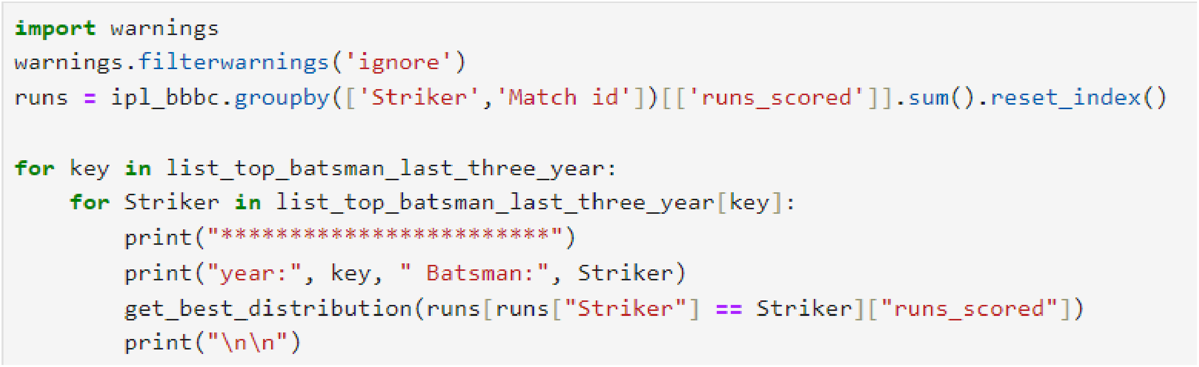
```
list_top_batsman_last_three_year
```

Result of top 3 runs scorer's



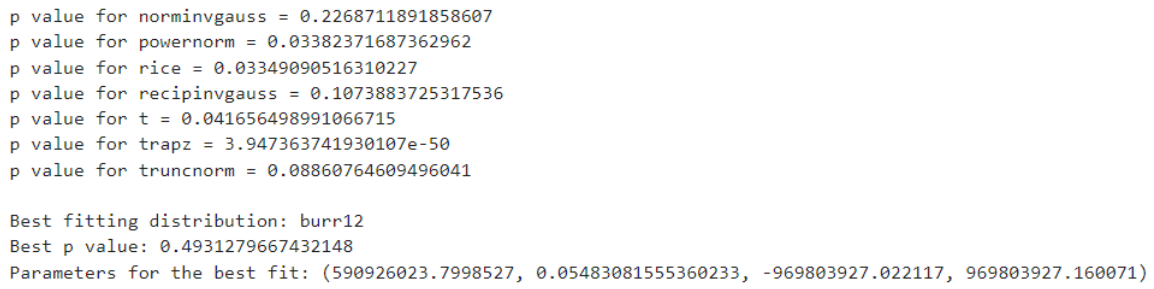
```
{2024: ['RD Gaikwad', 'V Kohli', 'B Sai Sudharsan'],
 2023: ['Shubman Gill', 'F du Plessis', 'DP Conway'],
 2022: ['JC Buttler', 'KL Rahul', 'Q de Kock']}
```

Codes for wickets taken:



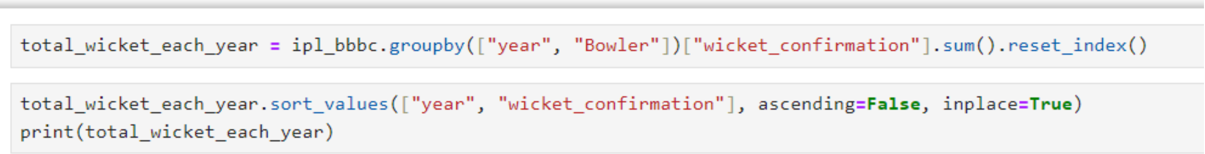
```
import warnings
warnings.filterwarnings('ignore')
runs = ipl_bbbc.groupby(['Striker','Match id'])[['runs_scored']].sum().reset_index()

for key in list_top_batsman_last_three_year:
    for Striker in list_top_batsman_last_three_year[key]:
        print("*************************")
        print("year:", key, " Batsman:", Striker)
        get_best_distribution(runs[runs["Striker"] == Striker]["runs_scored"])
        print("\n\n")
```

Result for wickets taken:



```
p value for norminvgauss = 0.2268711891858607
p value for powernorm = 0.03382371687362962
p value for rice = 0.03349090516310227
p value for recipinvgauss = 0.1073883725317536
p value for t = 0.041656498991066715
p value for trapz = 3.947363741930107e-50
p value for truncnorm = 0.08860764609496041

Best fitting distribution: burr12
Best p value: 0.4931279667432148
Parameters for the best fit: (590926023.7998527, 0.05483081555360233, -969803927.022117, 969803927.160071)
```

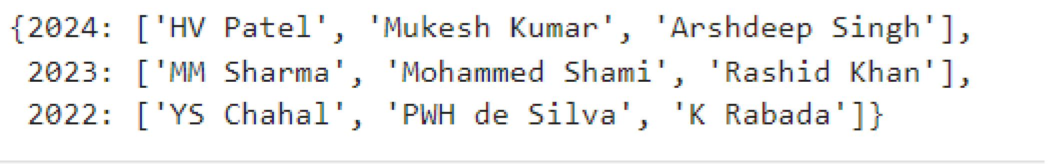
Codes for find best 3 ballers:



```python
total_wicket_each_year = ipl_bbbc.groupby(["year", "Bowler"])["wicket_confirmation"].sum().reset_index()
```

```python
total_wicket_each_year.sort_values(["year", "wicket_confirmation"], ascending=False, inplace=True)
print(total_wicket_each_year)
```

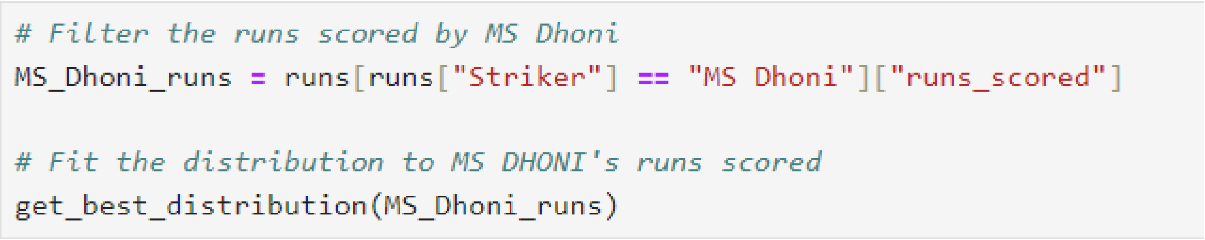
Result



```
{2024: ['HV Patel', 'Mukesh Kumar', 'Arshdeep Singh'],
 2023: ['MM Sharma', 'Mohammed Shami', 'Rashid Khan'],
 2022: ['YS Chahal', 'PWH de Silva', 'K Rabada']}
```

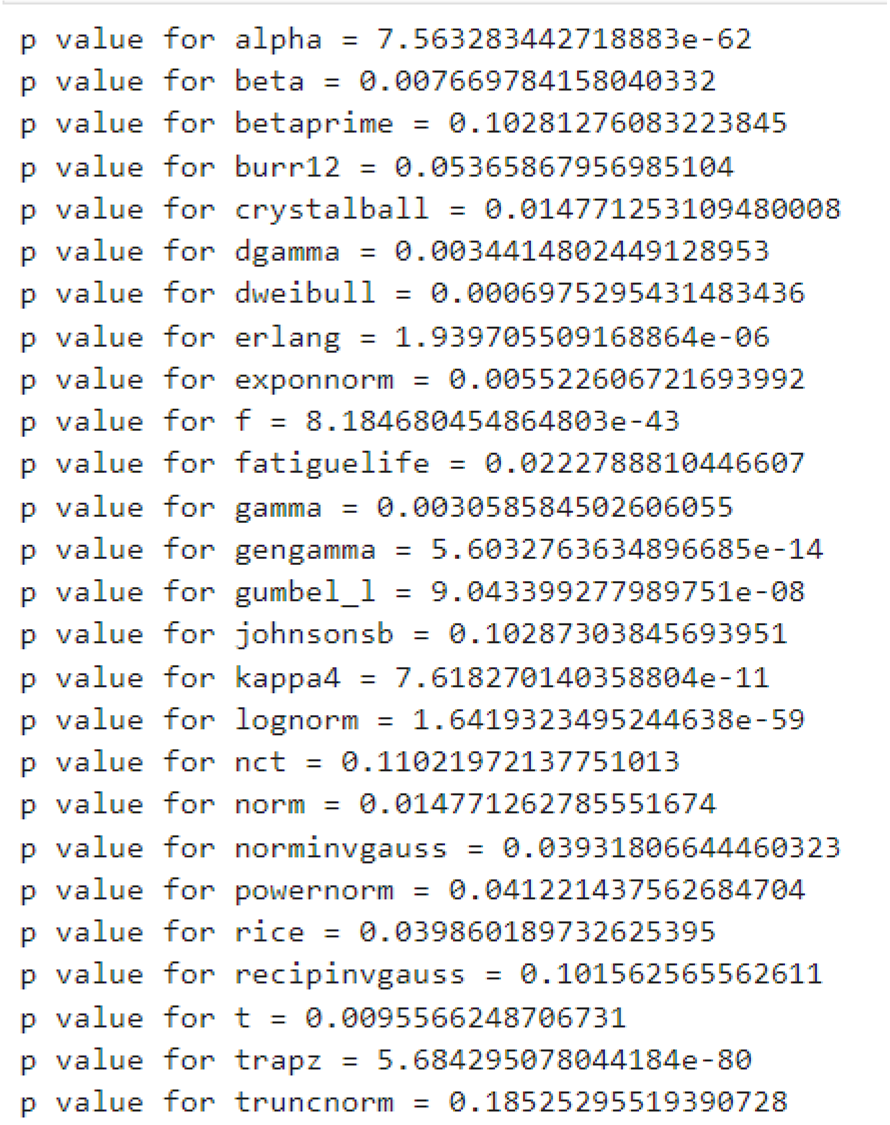
Runs scored by assigned player MS Dhoni

Code:

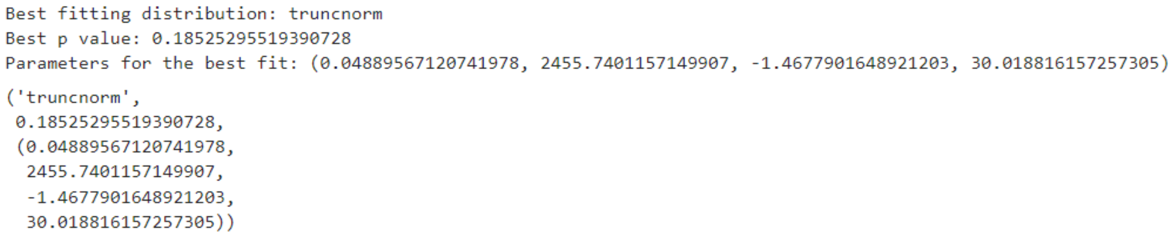


```python
# Filter the runs scored by MS Dhoni
MS_Dhoni_runs = runs[runs["Striker"] == "MS Dhoni"]["runs_scored"]

# Fit the distribution to MS DHONI's runs scored
get_best_distribution(MS_Dhoni_runs)
```

Result



```
p value for alpha = 7.563283442718883e-62
p value for beta = 0.007669784158040332
p value for betaprime = 0.10281276083223845
p value for burr12 = 0.05365867956985104
p value for crystalball = 0.014771253109480008
p value for dgamma = 0.0034414802449128953
p value for dweibull = 0.0006975295431483436
p value for erlang = 1.939705509168864e-06
p value for exponnorm = 0.005522606721693992
p value for f = 8.184680454864803e-43
p value for fatiguelife = 0.0222788810446607
p value for gamma = 0.003058584502606055
p value for gengamma = 5.6032763634896685e-14
p value for gumbel_l = 9.043399277989751e-08
p value for johnsonsb = 0.10287303845693951
p value for kappa4 = 7.618270140358804e-11
p value for lognorm = 1.6419323495244638e-59
p value for nct = 0.11021972137751013
p value for norm = 0.014771262785551674
p value for norminvgauss = 0.03931806644460323
p value for powernorm = 0.041221437562684704
p value for rice = 0.039860189732625395
p value for recipinvgauss = 0.101562565562611
p value for t = 0.0095566248706731
p value for trapz = 5.684295078044184e-80
p value for truncnorm = 0.18525295519390728
```



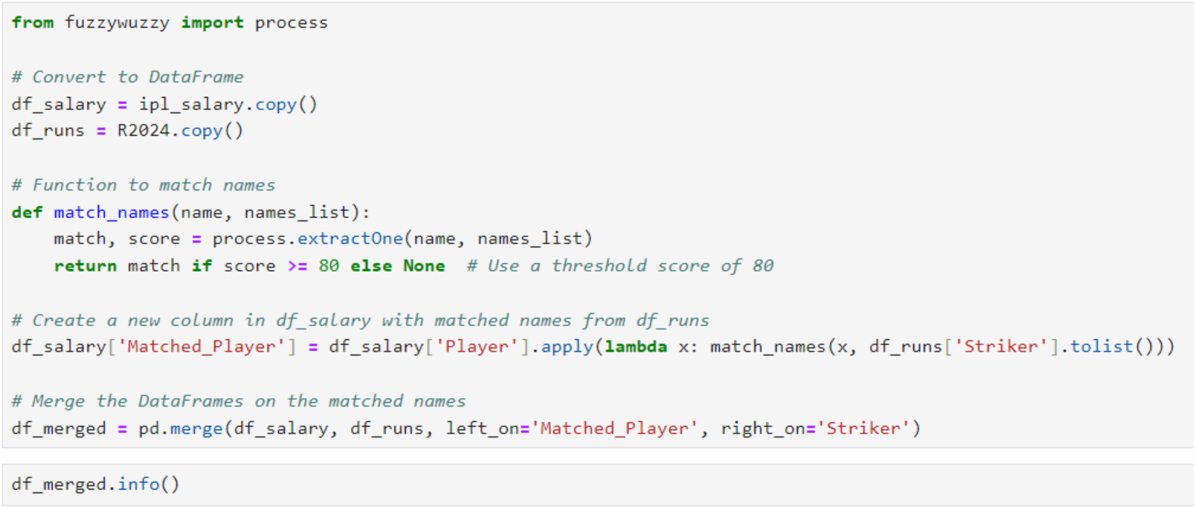
```
Best fitting distribution: truncnorm
Best p value: 0.18525295519390728
Parameters for the best fit: (0.04889567120741978, 2455.7401157149907, -1.4677901648921203, 30.018816157257305)
('truncnorm',
 0.18525295519390728,
 (0.04889567120741978,
  2455.7401157149907,
  -1.4677901648921203,
  30.018816157257305))
```

Interpretation

Finding the best bowlers wouldn't necessarily involve a separate distribution analysis. We could simply rank them based on the total number of wickets taken in the lost tournaments. 'HV Patel', 'Mukesh Kumar', 'Arshdeep Singh' are the best ballers in 2024 and RD Gaikwad', 'V Kohli', 'B Sai Sudharsan are the best bats man in 2024

- **Find the relationship between a player's performance and the salary he gets in your data.**

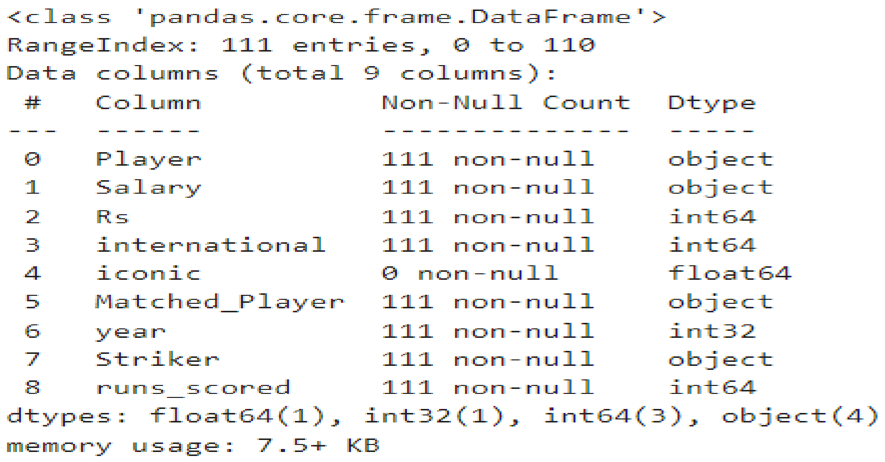
Codes:



```
from fuzzywuzzy import process

# Convert to DataFrame
df_salary = ipl_salary.copy()
df_runs = R2024.copy()

# Function to match names
def match_names(name, names_list):
    match, score = process.extractOne(name, names_list)
    return match if score >= 80 else None  # Use a threshold score of 80

# Create a new column in df_salary with matched names from df_runs
df_salary['Matched_Player'] = df_salary['Player'].apply(lambda x: match_names(x, df_runs['Striker'].tolist()))

# Merge the DataFrames on the matched names
df_merged = pd.merge(df_salary, df_runs, left_on='Matched_Player', right_on='Striker')
```

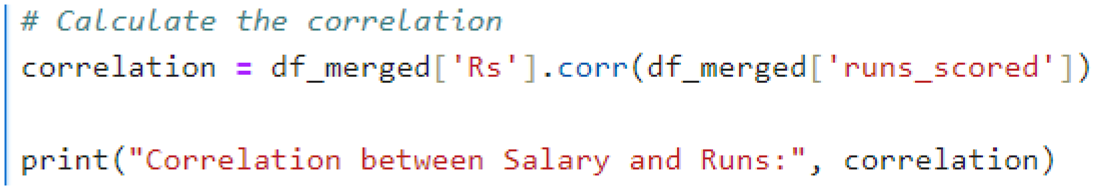
```
df_merged.info()
```

Result:



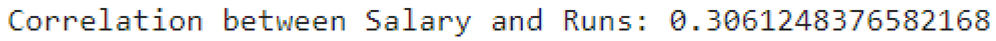
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 111 entries, 0 to 110
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Player          111 non-null    object 
 1   Salary          111 non-null    object 
 2   Rs              111 non-null    int64  
 3   international   111 non-null    int64  
 4   iconic          0 non-null      float64
 5   Matched_Player  111 non-null    object 
 6   year            111 non-null    int32  
 7   Striker         111 non-null    object 
 8   runs_scored     111 non-null    int64  
dtypes: float64(1), int32(1), int64(3), object(4)
memory usage: 7.5+ KB
```

Calculate the correlation of player salary and runs scored:



```
# Calculate the correlation
correlation = df_merged['Rs'].corr(df_merged['runs_scored'])

print("Correlation between Salary and Runs:", correlation)
```

Result:



```
Correlation between Salary and Runs: 0.3061248376582168
```

Interpretation:

The correlation coefficient of 0.3 indicates a weak positive correlation between player salary and runs scored. This means that there is a slightly positive trend, where players with higher salaries tend to also have scored more runs. However, the correlation is weak, so it's not a very strong relationship. Correlation doesn't imply causation. Just because players with higher salaries tend to score more runs doesn't necessarily mean that a higher salary causes a player to perform better. There could be other factors at play, such as more experienced players getting both higher salaries and scoring more runs. This analysis likely only considers runs scored as a metric for player performance. A more comprehensive analysis might consider other performance metrics relevant to the sport.

Reference

Web site W3 school : https://www.w3schools.com/python/