My title*

My subtitle if needed

Siddharth Gowda

November 30, 2024

First sentence. Second sentence. Third sentence. Fourth sentence.

1 Introduction

*Code and data are available at: https://github.com/siddharthgowda/ipl_wicket_analysis.

2 Data

Table 1: All Variables in the Cleaned Data Set

Match ID	Over	Striker	Bowler	Runs Off Bat	Wicket	Run Rate	Batting Style	Bowling Style	Previous Over Wickets
1254058	1	RG Sharma	Mohammed Siraj	0	FALSE	6	Right hand Bat	Right arm Fast	0

Table 1 show all data sample of a row in the cleaned data set. While there technically are more variables in the data set, these are the most important variables in the data set. Information about all variables in the cleaned data set are visible in Section D, including variables not in Table 1.

2.1 Measurement

All cricket data is from the (R Core Team 2023) package (**cricketdata?**). The data that was used is from the CRAN version of the package and not the (**github?**) developer tools version because the CRAN is more stable. (**cricketdata?**) takes data from (**cricinfo?**) and (**cricsheet?**). While not stated by (**cricketdata?**), based on the code the team behind the package are scrapping the HTML of ESPN Cricinfo and Cricsheet, As a result, anyone using the package must respect the rate limits set by these sites. Therefore, the data was first saved as CSV before performing any other cleaning and analysis.

More information about the raw data sets, measurements, (**cricsheet?**), (**cricinfo?**), and (**cricketdata?**) is in Section F.

2.2 Data Cleaning

In terms of data cleaning, only data from (**ipl?**) season from 2021 to 2024 (the most recent tournament) is present. This was none to make sure all data analysis about the IPL cricket is up to date, as certain strategies that used to be effective are no longer effective in IPL cricket. Also, two data sets were used IPL men's tournament data from 2021 to 2024 and player meta data (more about this in Section F). The IPL men's data set had play-by-play data for each ball bowled in each game in the IPL season. The two data sets were merged so that striker meta data and bowler meta data were included in each line. This including things like batting style and bowling style. Moreover, certain rows with missing data were removed and the current run rate and number of wickets in the previous over variables were

created based variables orginally included in the raw data set. Also, some data rows had clearly incorrect values, like a innings value of 6, when there are only 2 innings in T20 IPL cricket. These rows were removed from analysis.

2.3 Estimand and Response Variable

The estimated for the research paper is likelihood a ball will result in a wicket. The goal of the paper is to understand when wickets occur. The wicket variable is the response variable.

2.4 Predictor Variables

All other variables besides wicket listed in the begging of Section 2 are predictor variables.

Table 2: Fast Bowlers Are More Common Then Spinners

Bowling Style	Number of Bowlers
Left arm Fast	4
Left arm Fast medium	13
Left arm Medium	6
Left arm Medium fast	9
Left arm Wrist spin	4
Legbreak	17
Legbreak Googly	11
Right arm Fast	25
Right arm Fast medium	28
Right arm Medium	29
Right arm Medium fast	22
Right arm Offbreak	27
Slow Left arm Orthodox	23

Based on Table 2, there are more non-spin bowlers than spin bowlers. Spin bowlers include wrist spin, legbreak, and legbreak googly, and pace bowlesr are fast, medium fast, and medium. Also, there are more right arm bowlers than left arm bowlers.

Table 3: There are more right hand batsman.

Batting Style	Number of Batters
Left hand Bat	83
Right hand Bat	207

From Table 3, way more batsman are right handed than left handed.

2.5 Relationship Between Wickets and other Variables

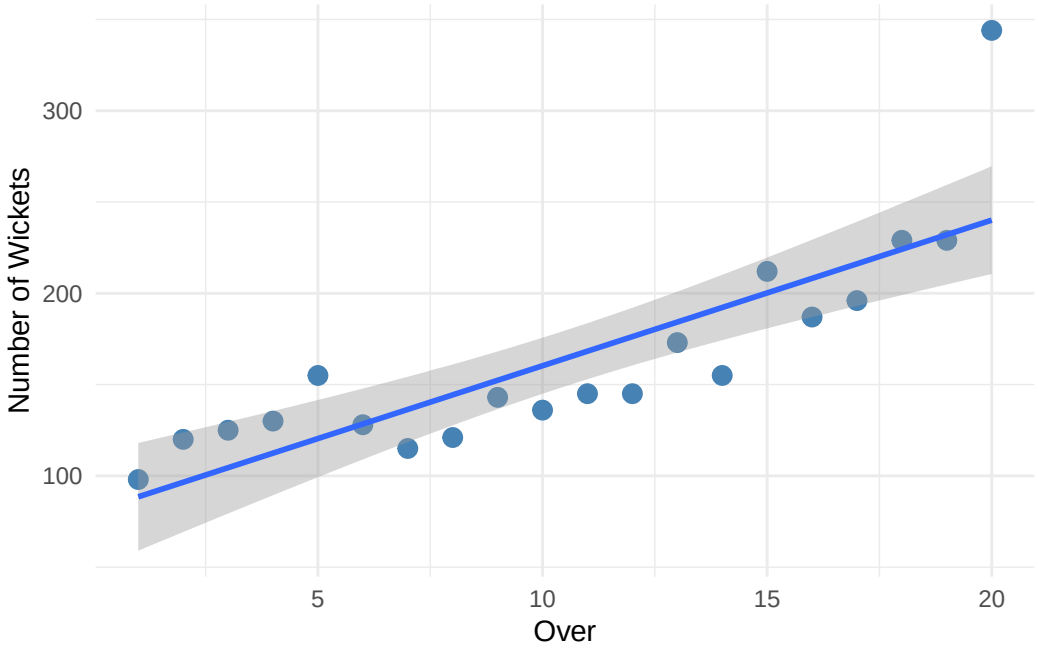


Figure 1: Wickets are more likely later in the innings

From Figure 1, there is a pretty strong linear positive relationship between the over number and the number of wickets that occur. It is important to note that in the last over (over 20) wicket occur way more than other overs. Also over 5 has more wickets that linear trend.

Based on Figure 2, for left hand batters, left arm fast bowlers have the highest chance for wickets. For right hand batters, medium pace or medium fast pace bowlers are the most likely to get wickets.

In Figure 3, there is a strong positive correlation between the average number of wickets in the previous over and the number of wickets in the current over. It is important to note that there is a gap in data for 0.5 average wickets in previous over to 0.8. Thus, it is difficult to say if from 0.5 to 0.8 the trend is still positive and linear.

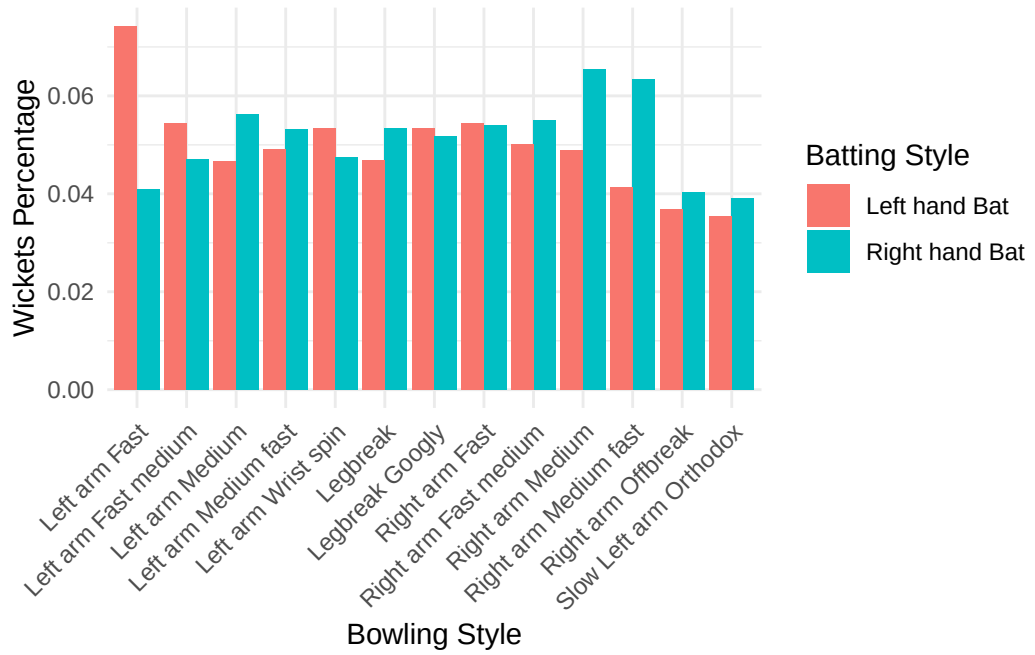


Figure 2: Wickets are more likely later in the innings

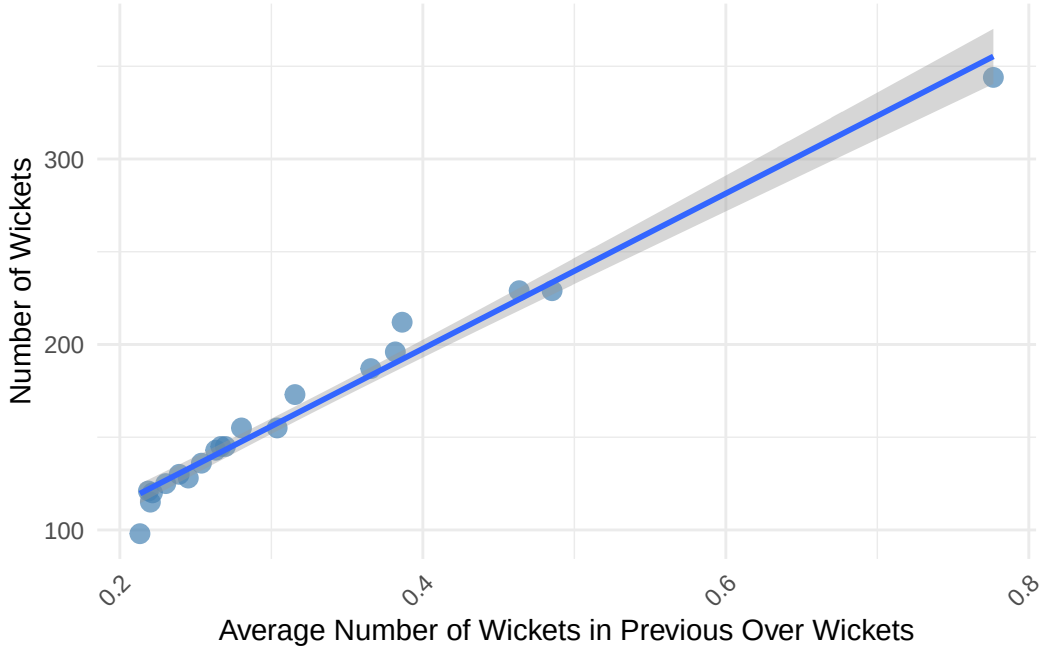


Figure 3: Correlation Between Wickets in the Last and Next Over

Table 4: Logistic Regression Model Ouput for Wicket Prediction Using Only the Over

	Wicket Prediction (Single Variable Input)
(Intercept)	−3.569
	(0.048)
over	0.058
	(0.004)
Num.Obs.	51 587
AIC	20 589.7
BIC	20 607.4
Log.Lik.	−10 292.858
RMSE	0.22

3 Model

The goal of the model is accurately predict when a wicket will occur in an (**ipl?**) game based on relevant variables from the dataset. Furtherore, I wanted to create a model that would achieve the most accuracy with the least amount of variables. To achieve this, three different generalized binomial family linear models (logistic regression) were used. The first model only had one input variable, the second one had two input variables, and third had four input variables. All models had the wicket boolean variable as the response variable.

3.1 Model Set-Up for Single Variable Generalized Linear Model

The model is set up using the `glm` (R Core Team 2023) CRAN package. It uses binomial family with the specified formula.

$\Pr(\text{Wicket} = 1) = \text{logit}^{-1}(\beta_0 + \beta_1 \cdot \text{over})$

Beta Explaiation:

β_0: Represents the general intercept, which represents the log-odds of a wicket occuring when the over variables does not matter

β_1: Represents the coefficent for the current over number

Based on Table 4, modeling with only the over variable accounts 0.22 RSME variability. Also for an increase in one over, the log odds of a wicket occuring increase by 0.058213 with a be value that is almost zero (less than $2 * 10^{-}16$). In terms of deviance, while not shown in the table, residual deviance of the model is roughly 300 degrees of freedom less than the null. The

Table 5: Logistic Regression Model Ouput for Wicket Prediction Using Over and Previous Wicket

	Wicket Prediction (Two Variable Input)
(Intercept)	−4.139
	(0.052)
over	0.005
	(0.004)
prev_over_wickets	1.852
	(0.031)
Num.Obs.	51 587
AIC	16 590.1
BIC	16 616.7
Log.Lik.	−8292.074
RMSE	0.21

null deviance is 20849 on 51586 degrees of freedom while the residual deviance is 20586 on 51585 degrees of freedom.

3.2 Model Set-Up for Two Variable Variable Generalized Linear Model

The model is set up using the `glm` (R Core Team 2023) CRAN package. It uses binomial family with the specified formula.

$\Pr(\text{Wicket} = 1) = \text{logit}^{-1}(\beta_0 + \beta_1 \cdot \text{over} + \beta_2 \cdot \text{previousOverWickets})$

β_0: Represents the general intercept, which is the value that occurs when all other variables cancel each other out

β_1: Represents the coefficent for the current over number

β_2: Represents the coefficent for the number of wickets in the previous over

Based on Table 5, modeling with only the over variable accounts 0.22 RSME variability. Also for an increase in number previous wickets in the previous over, the log odds of a wicket occuring increase by 1.852 with a p value that is almost zero (less than $2 * 10^{-}16$). The null deviance is 20849 on 51586 degrees of freedom while the residual deviance is 16584 on 51584 degrees of freedom. The logs odds increase for over is 0.005 with a p value of 0.168.

3.3 Model Set-Up for Four Variable Generalized Linear Model

Adding the two extra variables, batting style and bowling style, to the model did not change the effecitveness in predicability at all. More rigouris analysis avaliable in Section A.

3.4 Model justification

Based on Figure 1, there was a strong positive linear relationship between the over and the number of wickets taken, suggestign that the over can be a predictor of a wicket occuring. From Figure 2 certain matchup produced a higher chance of a wicket occuring, such as left hand fast on a left hand batter, which is why both variables were added to the four variable model in Section A. Finally from Figure 3, the average number of wickets that occured in the previous over had a strong positive linear relationship with the number of wickets that occured in balls in the next over, implying that previous over wickets impact the wicket probability of a ball in the next over.

4 Results

4.1 One Variable Model

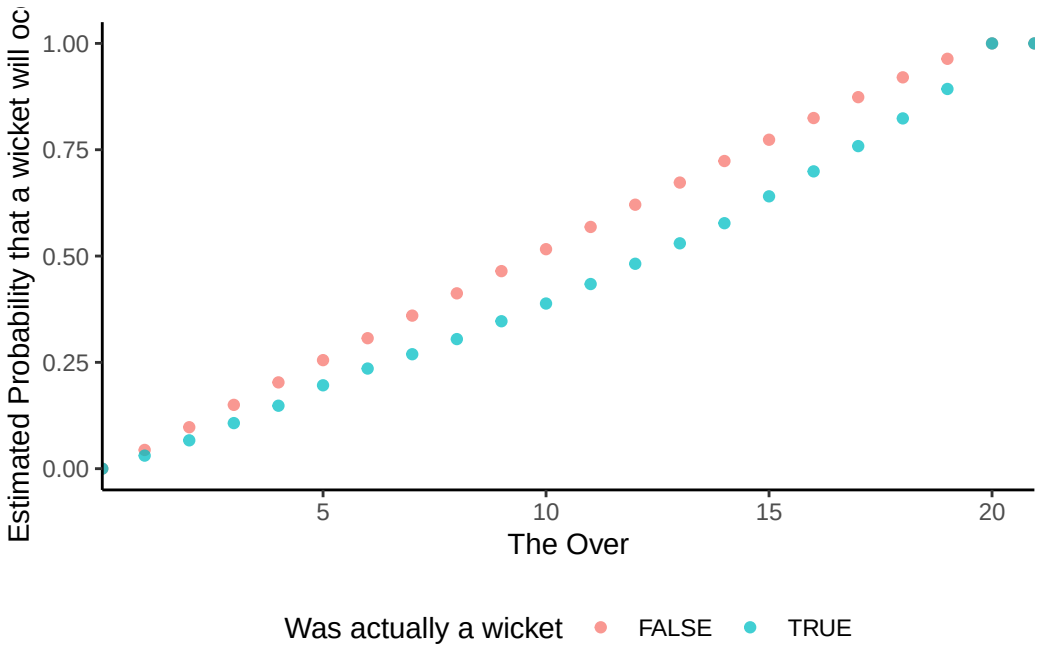


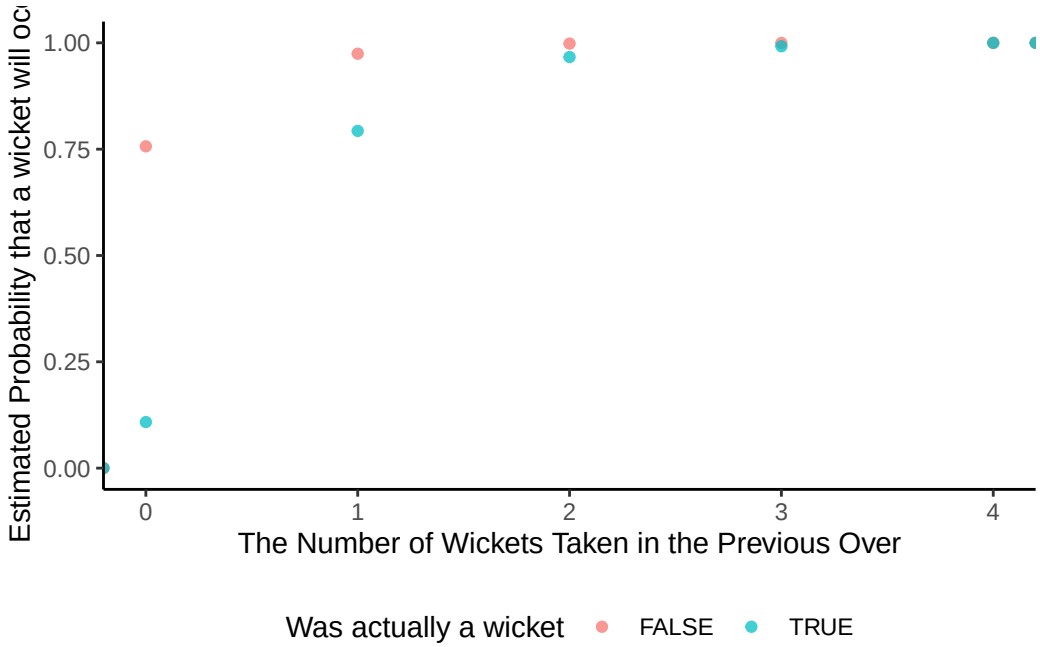
Figure 4: Correlation Between Wickets in the Last and Next Over

```r
test_data_simple <- test_data
predictions <- predict(simple_glm_wicket_model, newdata = test_data_simple, type = "response"

test_data_simple$predicted_wicket_prob <- predictions
test_data_simple <- test_data_simple %>%
  mutate(predicted_wicket = predicted_wicket_prob >= 0.5) %>%
  mutate(correct_prediction = predicted_wicket == wicket)

summary_results <- test_data_simple %>% group_by(wicket) %>%
  summarise(
  correct = sum(correct_prediction),
  incorrect = sum(!correct_prediction)
)

summary_results
```

```
# A tibble: 2 x 3
  wicket correct incorrect
  <lgl>    <int>     <int>
1 FALSE    12253         0
2 TRUE         0       644
```

4.2 Two Variable Model

```r
complex_glm_wicket_model_predictions <-
  predictions(complex_glm_wicket_model) |>
  as_tibble()

complex_glm_wicket_model_predictions |>
  mutate(wicket = factor(wicket)) |>
  ggplot(aes(x = prev_over_wickets, y = estimate, color = wicket)) +
  stat_ecdf(geom = "point", alpha = 0.75) +
  labs(
    x = "The Number of Wickets Taken in the Previous Over",
    y = "Estimated Probability that a wicket will occur",
    color = "Was actually a wicket"
  ) +
  theme_classic() +
  theme(legend.position = "bottom")
```




```
test_data_complex <- test_data
predictions <- predict(complex_glm_wicket_model, newdata = test_data_complex, type = "respons

test_data_simple$predicted_wicket_prob <- predictions
test_data_simple <- test_data_simple %>%
  mutate(predicted_wicket = predicted_wicket_prob >= 0.5) %>%
  mutate(correct_prediction = predicted_wicket == wicket)

summary_results <- test_data_simple %>% group_by(wicket) %>%
  summarise(
  correct = sum(correct_prediction),
  incorrect = sum(!correct_prediction)
)

summary_results
```

```
# A tibble: 2 x 3
  wicket correct incorrect
  <lgl>    <int>     <int>
1 FALSE    12229        24
2 TRUE        17       627
```

4.3 Four Variable Model

5 Discussion

5.1 First discussion point

If my paper were 10 pages, then should be be at least 2.5 pages. The discussion is a chance to show off what you know and what you learnt from all this.

5.2 Second discussion point

Please don't use these as sub-heading labels - change them to be what your point actually is.

5.3 Third discussion point

5.4 Weaknesses and next steps

Weaknesses and next steps should also be included.

Appendix

A Four Variable Model

The model is set up using the `glm` (R Core Team 2023) CRAN package. It uses binomial family with the specified formula.

$$\Pr(\text{Wicket} = 1) = \text{logit}^{-1}\left(\beta_0 + \beta_1 \cdot \text{over} + \beta_2 \cdot \text{previousOverWickets} + \beta_3 \cdot \text{BowlingStyle} + \beta_4 \cdot \text{BattingStyle}\right)$$

β_0: Represents the general intercept, which is the value that occurs when all other variables cancel each other out.

β_1: Represents the coefficent for the current over number

β_2: Represents the coefficent for the number of wickets in the previous over

β_3: Represents the coefficent for the bowling style. Each bowling style has its own coefficent.

β_4: Represents the coefficent for the batting style (left handed or right handed). Each batting style has its own coefficent.

Based on Table 6, none of the batting styles or bowling style coefficent are statistically significant all with p values greater than 10%. This implies that most likely these variables not great predictors of wicket creation. The number of wickets in the previous over still has a statistically signifcant impact and has an increase of 0.031446 log odds per an increase of one wicket in the previous over. The residual deviance is 16574 on 51571 degrees of freedom compared to the null of 20849 on 51586 degrees of freedom.

B Variables Not Part of Model

```
# A tibble: 6 x 20
  match_id  year venue     innings  over  ball batting_team bowling_team striker
     <dbl> <dbl> <chr>       <dbl> <dbl> <dbl> <chr>        <chr>        <chr>
1  1254058  2021 MA Chida~       1     1     2 Mumbai Indi~ Royal Chall~ RG Sha~
2  1254058  2021 MA Chida~       1     1     3 Mumbai Indi~ Royal Chall~ RG Sha~
3  1254058  2021 MA Chida~       1     1     4 Mumbai Indi~ Royal Chall~ RG Sha~
4  1254058  2021 MA Chida~       1     1     5 Mumbai Indi~ Royal Chall~ RG Sha~
5  1254058  2021 MA Chida~       1     1     6 Mumbai Indi~ Royal Chall~ RG Sha~
6  1254058  2021 MA Chida~       1     2     1 Mumbai Indi~ Royal Chall~ RG Sha~
# i 11 more variables: bowler <chr>, runs_off_bat <dbl>,
#   wickets_lost_yet <dbl>, wicket <lgl>, target <dbl>, run_rate <dbl>,
#   batting_style <chr>, batter_playing_role <chr>, bowling_style <chr>,
#   bowler_playing_role <chr>, prev_over_wickets <int>
```

C Extra Tables

```
bowling_batting_role_matchup_boundaries <- cleaned_data %>%
  group_by(bowling_style, batter_playing_role) %>%
  summarise(
    wickets_percentage = round(sum(wicket == TRUE)/n(), 3),
    num_balls = n(),
  ) %>% arrange(desc(wickets_percentage), bowling_style, batter_playing_role)
```

```
`summarise()` has grouped output by 'bowling_style'. You can override using the
`.groups` argument.
```

```
bowling_batting_role_matchup_boundaries
```

Table 6: Batting Style vs Bowling Style Do Not Heavily Impact Wicket Potential

	(1)
(Intercept)	−4.316
	(0.192)
over	0.005
	(0.004)
prev_over_wickets	1.853
	(0.031)
batting_styleRight hand Bat	0.044
	(0.045)
bowling_styleLeft arm Fast medium	0.099
	(0.207)
bowling_styleLeft arm Medium	0.255
	(0.212)
bowling_styleLeft arm Medium fast	0.205
	(0.206)
bowling_styleLeft arm Wrist spin	0.158
	(0.235)
bowling_styleLegbreak	0.257
	(0.212)
bowling_styleLegbreak Googly	0.207
	(0.196)
bowling_styleRight arm Fast	0.190
	(0.192)
bowling_styleRight arm Fast medium	0.211
	(0.197)
bowling_styleRight arm Medium	0.109
	(0.193)
bowling_styleRight arm Medium fast	0.165
	(0.204)
bowling_styleRight arm Offbreak	0.104
	(0.200)
bowling_styleSlow Left arm Orthodox	0.011
	(0.202)
Num.Obs.	51 587
AIC	16 605.8
BIC	16 747.4
Log.Lik.	−8286.914
RMSE	0.21

14

Figure 5: Some stadiums are more prone to wickets than others.

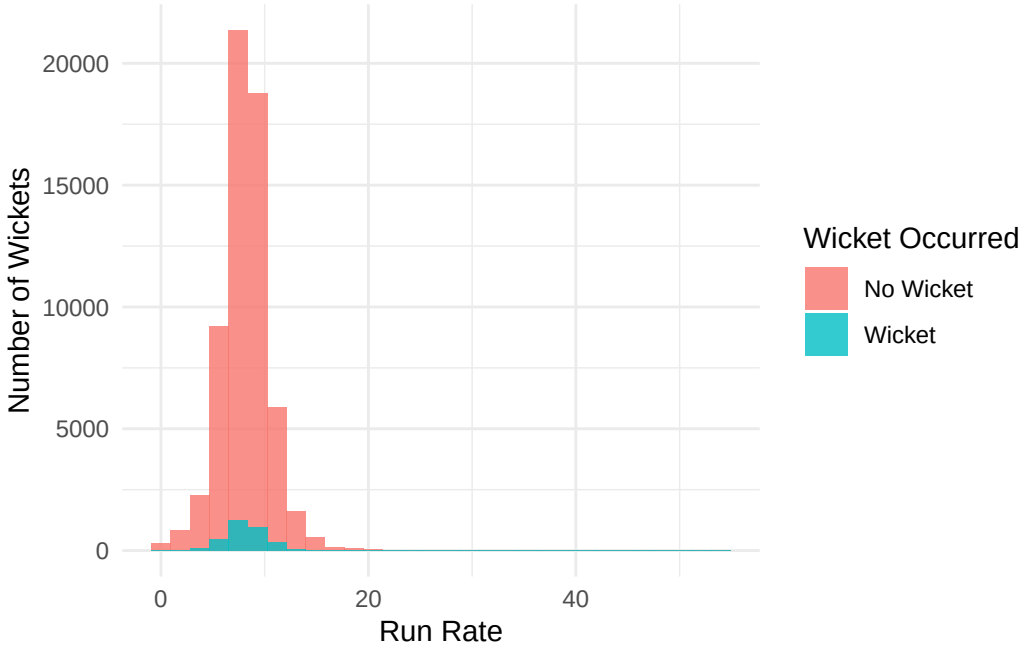


Figure 6: Current Run Rate Does Not Impact Wicket Rate

```
# A tibble: 117 x 4
# Groups:   bowling_style [13]
   bowling_style         batter_playing_role wickets_percentage num_balls
   <chr>                 <chr>                            <dbl>     <int>
 1 Left arm Fast medium  Bowling Allrounder               0.121       140
 2 Right arm Fast        Bowler                           0.112       489
 3 Left arm Fast         Bowling Allrounder               0.111        18
 4 Left arm Fast         Bowler                           0.108        37
 5 Left arm Medium       Bowling Allrounder               0.108        83
 6 Left arm Fast medium  Bowler                           0.104       154
 7 Right arm Medium      Bowler                           0.101       671
 8 Right arm Medium fast Bowler                           0.09        222
 9 Left arm Fast         Allrounder                       0.087       115
10 Left arm Wrist spin   Bowler                           0.086        81
# i 107 more rows
```

D Cleaned Data Dictionary

Below are all of the variables in the cleaned data with an explanation of each variable, examples, and the data type.

Match id (Integer)
Example: 335982
A randomly generated ID uniquely identifies each cricket match in the data set.

Year (Integer)
Example: 2022
The year the match was played in.

Venue (string)
Example: M Chinnaswamy Stadium
The location the match is played in.

Innings (Integer)
Example: 1
The innings the current ball is played in. Should only be either 1 or two.

Over (Integer)
Example: 7
The over the current ball is played in. This should be from 1 to 20 in T20 Cricket.

Ball (Integer)
Example: 3
The current ball number in the over. This typically should be from 1 to 6, but could be more if no balls or wides are given.

Batting Team (String)
Example: Kings XI Punjab
The team that is currently batting.

Bowling Team (String)
Example: Mumbai Indians
The team that is currently bowling.

Striker (String)
Example: AJ Finch
The player that is currently on the wicket the bowler is bowling to.
Bowler (String)
Example: SMSM Senanayake
The player that is bowling.

Runs off of the bat (Integer)
Example: 3
The number of the runs the team scored from this ball directly off of the striker's bat.

Wicket Lost Yet (Boolean)
Example: True
Did a wicket occur in this ball.
Wickets Lost (Integer)
Example: 7
The number of wickets that have fallen. From 0 to 10.

Target (Integer)
Example: 207
The par course for the given venue's pitch on that match's day.

Run Rate (Float)
Example: 6.50
The current run of the batting team in the match. This is the number of runs the team scores per over.

Batting Style (String)
Example: Left hand Bat
The batting style of a player. This is either left hand or right hand.

Bowling Style (String)
Example: Slow Left arm Orthodox
The bowling style of a player. This only exists if a player is a bowler.

Batting Player Role (String)
Example: Allrounder
The role of a batting player on the team.

Bowling Player Role (String)
Example: Bowler
The role of a bowling player on the team.

Previous Over Wickets (Number)
Example: 2
The number of wickets the bowling team had in the previous over.

E Idealized Methodology

This section will go over a comprehensive plan that could be used to translate real time cricket plays to actual data sets and databases that can be accessed by individuals interested in performing statistical analysis. Since this data is not a survey and is not really sampling

based, there will be no mention of terms like stratified or cluster sampling or selection and response bias. This example will only be about the (**ipl?**) level cricket since this paper is only about the (**ipl?**). However, most of this methodology can be used in other cricket leagues and other formats.

E.1 Translating Plays to Data

The process of converting live cricket plays into structured data begins with real-time observation and documentation. During an Indian Premier League (IPL) match, a dedicated data entry team should closely monitor every moment of the game, capturing critical information about each play as it unfolds. A primary data collector focuses on immediate, live-capturable elements such as the bowler's name, the batsman on strike, the non-striking batsman, the type of bowling delivery, number of runs scored, and any fielding actions. This real-time data collection requires trained personnel with a deep understanding of cricket's nuanced rules and scoring mechanisms.

Simultaneously, a secondary team prepares for post-match data verification and enrichment. After the live match, this team reviews video recordings, carefully cross-referencing the initial data entries to correct any potential errors or omissions. This post-match review allows for more precise data collection, particularly for complex plays that might have been challenging to capture in real-time. The verification process involves multiple checks, including reviewing ball-tracking technologies, replay footage, and official match scorecards to ensure absolute accuracy.

The data entry process is designed with multiple layers of validation. Trained data entry specialists use specialized software that provides immediate validation checks, flagging potential inconsistencies or unusual entries for immediate review. This approach minimizes human error and ensures the highest possible data integrity. The entire process is structured to capture not just the basic outcomes of plays, but the rich contextual information that makes cricket data valuable for deeper statistical analysis.

E.2 Making Data Easy to Use

To make this data easy to use, the team should create a clean structured or SQL database. Creating a robust and user-friendly database requires careful architectural planning. The proposed database design incorporates multiple interconnected tables that provide a comprehensive view of IPL cricket data. A player table serves as a central repository, with each player assigned a unique identifier that can be referenced across other tables. This player's primary key ID becomes a critical link in tables such as play, match, and season tables, enabling complex queries and data relationships.

The database design prioritizes data reliability and efficiency. By using multiple tables, the system eliminates redundancy and minimizes storage requirements. For instance, a match

table would contain high-level match information, while a play table would capture granular, play-by-play details. A season table allows for broader and simple analysis, tracking changes and trends, winners, top scorers and more, across different IPL seasons. This structured approach not only improves data storage efficiency but also enhances the usability of the data set for researchers, analysts, and cricket enthusiasts.

E.3 Making Data Accessible

Data accessibility is fundamental to the value of this cricket data set. The proposed system implements a controlled access mechanism through a carefully designed API that allows broad yet secure data retrieval. An API key system ensures that access is regulated, preventing potential system overloads while maintaining open accessibility. Rate limiting mechanisms are strategically implemented to protect the system from potential denial-of-service attacks, ensuring consistent data availability for all authorized users.

The public API is deliberately designed as a read-only interface, preventing unauthorized modifications to the core data set. This approach maintains data integrity while still providing transparent access to researchers, analysts, and cricket enthusiasts. By making the data publicly accessible through a controlled API, the system inherently supports data validation. Multiple users can cross-reference and verify the data, creating a collaborative environment of data integrity and trust.

Overall, this methodology represents a comprehensive approach to transforming live observational cricket plays into a structured, accessible, and reliable data set that can support advanced statistical analysis and deeper understanding of the Indian Premier League.

F Data Sheet for The Raw Data

F.1 Questions

F.1.1 Who put the data set together?

The raw data set was acquired from the (**cricketdata?**) R package. This data set had many contributors which can be seen through this link, but the main developer is Rob Hyndman. The data set scraps the html from (**espncricinfo?**) and (**cricsheet?**). These are two reputable organizations when it comes to cricket game data. (**cricsheet?**) is moreso responsible for play by play cricket data, like giving the data for what happens for every ball in a game, whereas (**espncricinfo?**) is more responsible for general player career information and statistics.

F.1.2 Who paid for the data set to be created?

The (**cricketdata?**) R package is completely open source and free under the GPL-3 license. The source code can be seen through this link. In terms of ESPN Cricinfo, this is a site and organization that makes money talking about cricket and airing games. So, it is in their best interest to give basic statistics for free on their website to entice people to use their services to watch and interact with content related to cricket. Cricsheet is operated by Stephen Rushe. He has written code to extract play-by-play cricket match data, which most likely means he is web scraping. There is not much other information about the code that extracts the data and what sites he uses.

F.1.3 How complete is the data set? {#sec-complete-data set}

The play by play data is very complete. For each ball in a game, it includes the bowler, batter, the batter on the opposite wicket, the venue, teams, the runs given, and more. The important data that seems to be missing is data related to the position of the fielders and the distance and final location of the ball once and if it hits the bat. In terms of player info, it is also very complete. It includes information about their batting style, blowing style if applicable, and their role on the team. The only issue here is that a player's role on the team can change depending on the team they are one. For instance, a player might be an opening batsman when playing in test cricket but could be a midover batsman when playing in T20. This data set does not account for this discrepancy.

F.1.4 Which variables are present, and, equally, not present, for particular observations?

For play-by-play data the bowler, battsman, and teams involved in the play are present and so is general game information. Furthermore, the final outcome of the play, such as the number of runs, or if a wicket occurred are also given. Information about ball speed, distance, location, and fielding is not given. For player information, information about their country of citizenship, country of birth, batting style, bowling style, and batting order, and player role on a team are given. Potential missing observations with the player information are listed in **?@sec-complete-data** set.

F.2 Data Dictionary For Raw Data

For this report, there are only two data sets used, the IPL (Indian Premier League) play-by-play data and player metadata. The variables for these data sets will be listed here with examples.

F.2.1 Play-by-play IPL Data

Match id (Integer)
Example: 335982
A randomly generated ID uniquely identifies each cricket match in the data set.

Season (String)
Example: 2007/8 or 2024
Represents the IPL tournament year of the match. For instance "2007/8" represents the 2007 to 2008 IPL season and 2024 represents the tournament that was played only in 2024.

Season (Date)
Example: 2024-04-15
Represents the day the match was played. It is in a year-month-day format.

Venue (String)
Example: M Chinnaswamy Stadium
The location the match is played in.

Innings (Integer)
Example: 1
The innings the current ball is played in. Should only be either 1 or two.

Over (Integer)
Example: 7
The over the current ball is played in. This should be from 1 to 20 in T20 Cricket.

Ball (Integer)
Example: 3
The current ball number in the over. This typically should be from 1 to 6, but could be more if no balls or wides are given.

Batting Team (String)
Example: Kings XI Punjab
The team that is currently batting.

Bowling Team (String)
Example: Mumbai Indians
The team that is currently bowling.

Striker (String)
Example: AJ Finch
The player that is currently on the wicket the bowler is bowling to.

Non-Striker (String)
Example: A Ashish Reddy
The player that is batting but is on the opposition wicket.

Bowler (String)
Example: SMSM Senanayake
The player that is bowling.

Runs off of the bat (Integer)
Example: 3
The number of the runs the team scored from this ball directly off of the strikers bat.

Extra Runs (Integer)
Example: 1
The number of runs that the team scored that was not because of the striker's bat. This could be because of wides.

Extra Ball (Boolean)
Example: False
Was this ball an extra ball? For instance, this ball could be extra due to a wide.

Balls Remaining (Integer)
Example: 79
The number of balls remaining in the game. This is from 0 to 120.

Runs Scored (Integer)
Example: 101
The number of runs already scored by the batting team.

Wicket (Boolean)
Example: True
Did a wicket occur in this ball.
Wickets Lost (Integer)
Example: 7
The number of wickets that have fallen. From 0 to 10.

First Innings Total (Integer)
Example: 165
The total number of runs earned in the first innings.

Second Innings Total (Integer)
Example: 168
The total number of runs earned in the second innings.

Target (Integer)
Example: 207
The par course for the given venue's pitch on that match's day.

Wides (Integer)
Example: 4
The number of wides bowled in this over.

No Balls (Integer)
Example: 2
The number of no balls bowled in this over.

Byes (Integer)
Example: 4
Runs scored by byes in this ball. This ranges from 0 to 4.

Leg Byes (Integer)
Example: 2
Runs scored by leg byes in this ball. This ranges from 0-4.

Wicket Type (String)
Example: Bowled
If a wicket occurred, this column will describe how it happened. It includes bowled, stumped, and caught.

Player Dismissed (String)
Example: A Ashish Reddy
If a wicket occurred, this column will have the name of the player that was dismissed.

F.2.2 Player Metadata

Cricinfo Player id (Integer)
Example: 1175501
A randomly generated ID uniquely identifies each cricket player from Cricinfo data.

Cricsheet Player id (Alphanumeric)
Example: 21d38d47
A randomly generated ID uniquely identifies each cricket player from Cricinfo data.

Unique Name (String)
Example: Alkandari Abdulrahman
The unique name of the player that is stored in the (**cricketdata?**) package. All other tables will only use the unique name of a player and not the full name of the player.

Full Name (String)
Example: AGHM Alkandari Abdulrahman
The full name of a player.

Country (String)
Example: Zimbabwe
The country of citizenship for this player.

Date of Birth (Date)
Example: 1971-09-21
The day the player was born in a year-month-day format.

Birthplace (String)
Example: Nangrahar, Afghanistan
The city and sometimes country a player was born in.

Batting Style (String)
Example: Left hand Bat
The batting style of a player. This is either left hand or right hand.

Bowling Style (String)
Example: Slow Left arm Orthodox
The bowling style of a player. This only exists if a player is a bowler.

Player Role (String)
Example: Allrounder
The role of a player on the team.

G References

What to cite: - cricketdata - ESPNCricinfo - Cricsheet - All tidyverse packages used

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.