Improving Duckworth-Lewis: Statistical Methods for Revising Score Targets in Limited-Overs Cricket

Matthew Knowles

Department of Mathematics,
University of York,
United Kingdom

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Mathematics

Abstract

Write this last

Contents

Chapter 1

Background

1.1 Limited-Overs Cricket and the need for Duckworth-Lewis

In 1963, the cricketing calender in the UK had for the first time a different format of the game ammended to it. The "Gillette Cup" introduced a form of cricket wherein each team has 1 innings, lasting 50 overs. The idea was that this competition, coming to a conclusion within the space of a day, would increase spectator numbers and by extension, ticket sales.

However, given the tournament-based nature of this new competition, we have the natural need for a definitive result. This is something that is not always guarenteed in "first class" cricket, where draws are common. [1]. As such, in order to allow for a result to be determined when a game is cut short, the idea of target-revision was introduced. This meant that a team could be set a roughly equivalent run target off of a reducded number of overs that would still classify them as the winners.

1.2 Problems raised with DLS

foo

1.3 Project Aims

bar

1.4 Review of current Literature

We begin the look at literature on this topic with the paper published by F. Duckworth himself in 1998 [2]. In this paper, Duckworth proposes the "D/L" method based on 5 principles. HOweverm the issue with the sentence that appears after defining the function $Z(u, w)$. "Comercial confidentiality prevents the disclosure of the mathematical definitions of these functions". The claim is that these functions have been via experimentation. This however gives rise to the first issue: the first T20 game of cricket was not played until 2003. So clearly, D/L was not designed with T20 in mind, and so the functions derived from experementation and research will not be accurate for T20 cricket.

In 2004, a year after the first T20 matches were played, Duckworth and Lewis published another paper [1], in which they report that the table used for calculating the method can be employed by

[1] Note that "draw" and "tie" are not interchangable terminology in cricketing terms

Chapter 2

Data

2.1 Data Origin

The primary source of data for carrying out this project was downloaded from "cricsheet"[1] and stored locally on a private server. In total there are 2167 individual mathes of data. Each in a JSON format. These cover matches ranging from the 3^{rd} of January, 2004. Up to the 20^{th} of July, 2021.

2.2 Attributes

Each JSON file contains a considerable amount of metadata surrounding the match in question. Along with ball-by-ball data for the entire match. We have access to attributes such as the date, where the match was played, the entire teamsheet for both teams, who the officials were, who won- and by what margin, who won the toss; and many others.

We also have the ball-by-ball data. So for every ball bowled, it gives who were the striking and non-striking batsmen, how many runs were scored and how. It also details if a wicket was taken that ball, and how.

2.3 Pre-Processing

Many Python scripts were written to get data in an appropriate form for analysis.

[1] https://cricsheet.org/

Chapter 3

The DLS Method in Detail

We now look at the mathematics behind the D/L, and DLS methods. D/L being the original method, and DLS the method that Stern helped to revise. In the original paper, the authors state "Commercial confidentiality prevents the disclosure of the mathematical definitions of these functions." [2]. Which is naturally a slight problem for this, but what we can do is instead use sample values based on data we have to look at how these functions behave, so all is certainly not lost.

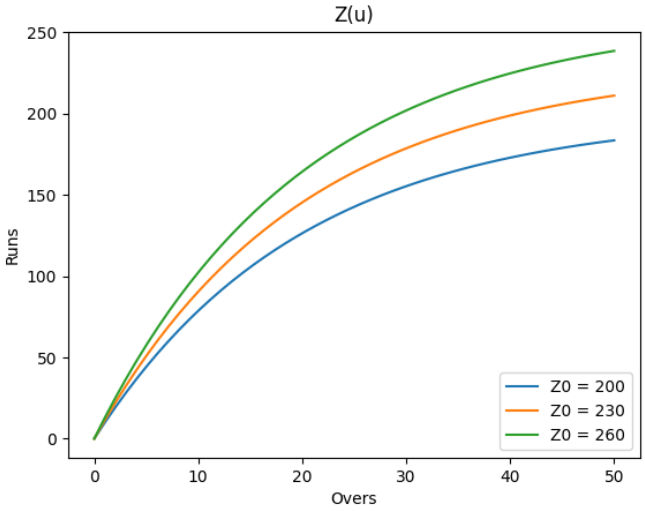
3.1 Origins: Duckworth and Lewis

We begin by looking at the original paper. The first thing to establish is how many runs are scored, on average, in a given number of overs. This is given by the equation:

$$Z(u) = Z_0[1 - exp(-bu)] \tag{3.1}$$

Where u is the number of overs, b is the exponential decay constant, and Z_0 is the average total score in first class cricket, but with one-day rules imposed.

Now because we don't have access to actual values for Z_0 or b, a plot to see what equation 3.1 looks like was created by using 3 sample values for Z_0. For b, it was a process of trial and error to find a value that resulted in the graph having a similar shape to original figures in the D/L paper.



However, we have not yet looked at what happens when wickets are lost. To introduce this

metric, equation 3.1 is revised to incorperate the scenario that w wickets have been lost, and that there are u overs remaining The revised equation is given as follows:

$$Z(u,w) = Z_0(w)[1 - exp(-b(w)u)] \tag{3.2}$$

Where now, we have $Z_0(w)$ giving the average total score from the last $10 - w$ wickets in first class cricket. We now also have $b(w)$ as the exponential decay constant, which now changes depending on wickets lost.

With this in mind, we now look at the specific case of equation 3.2 with $u = N$ and $w = 0$, namely, the conditions at the start of an N-over innings. We have:

$$Z(N,0) = Z_0[1 - exp(-bN)]. \tag{3.3}$$

Which we then incorperate into the ratio

$$P(u,w) = \frac{Z(u,w)}{Z(N,0)}. \tag{3.4}$$

The ratio 3.4 gives, keeping in mind there are u overs still to be bowled, with w wickets lost, the average proportion of the runs that still need to be scored in the innings. It is this ratio that is where the revised scores come from. Let us now look a bit more at how that works practically.

Example 3.1.1. *Assume there is a break in the second innings (due to rain or similar), which results in the second team missing some overs. Let u_1, u_2 be the number of overs played before the break, and available after it respectively. We impose the condition that $u_2 < u_1$. At the time of the break, the second team had lost w wickets. The aim is to adjst the required score to account for the $u_1 - u_2$ overs they have lost. The winning "resources" available are given by*

$$R_2 = [1 - P(u_1, w) + P(u_2, w)].$$

Which means, if the first team batting scored S runs, then the new tartget is given by

$$T = \lceil SR_2 \rceil$$

3.2 Improvements by Stern

foo

Chapter 4

Overview of Pattern Recognition Techniques

4.1 Model 1

Bar

Chapter 5

Implementing SPR Methods on Match Data

5.1 Method 1

Foo

Chapter 6

Analysis and Results

6.1 Method 1

foo

Chapter 7

Conclusion

foo

Bibliography

[1] FC Duckworth and AJ Lewis. A successful operational research intervention in one-day cricket. *Journal of the Operational Research Society*, 55(7):749–759, 2004.

[2] Frank C Duckworth and Anthony J Lewis. A fair method for resetting the target in interrupted one-day cricket matches. *Journal of the Operational Research Society*, 49(3):220–227, 1998.

Appendix A

Python Code for Chapter 2

Listing A.1: Data Wrangling Code

```
import json
import os
import csv

#Moves all the games from a set of files that were decided by DLS
def get_dls_results(inputPath, outputPath):
    os.chdir(inputPath)
    for f in os.listdir():
        with open(inputPath+f,"r") as file:
            data = json.load(file)
        if("method" in data["info"]["outcome"]):
            if(data["info"]["outcome"]["method"] == "D/L"):
                os.system("mv "+f+" "+outputPath)


def get_match_data(matchPath, csvPath):
    os.chdir(matchPath)
    for f in os.listdir():
        with open(matchPath+f, "r") as file:
            data = json.load(file)

        homeTeam = data["info"]["teams"][0]
        awayTeam = data["info"]["teams"][1]
        ground = data["info"]["venue"]

        if "winner" in data["info"]["outcome"]:
            winner = data["info"]["outcome"]["winner"]
            if "runs" in data["info"]["outcome"]["by"]:
                runsWonBy = data["info"]["outcome"]["by"]["runs"]
                wicksWonBy = 0
            else:
                wicksWonBy = data["info"]["outcome"]["by"]["wickets"
                    ]
                runsWonBy = 0
        else:
            winner = "tie"
```

```
        targetOvers = data["innings"][1]["target"]["overs"]
        targetRuns = data["innings"][1]["target"]["runs"]

        row = homeTeam,awayTeam,ground,winner,str(runsWonBy),str(
            wicksWonBy),str(targetRuns),str(targetOvers)
        csvF = open(csvPath, "a")
        writer = csv.writer(csvF)
        writer.writerow(row)
        csvF.close()


get_match_data("/Users/matthew/Documents/Uni/Dissertation/Data/dls/"
    , "/Users/matthew/Documents/Uni/Dissertation/Data/dlsMaster.csv")
```