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

Cricket is a game played by two teams, each with 11 players. The objective for the batting team is to score as many 'runs' as possible without losing 10 of their batsmen[1], who can be 'out' in a variety of ways. Cricket is played in 'overs', each over lasting 6 deliveries. Traditionally, the game lasted for 4 days and there was no limit to the number of overs the bowling team could bowl.

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. [2]. 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 [**?**]. 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

[1] There have to be two batters on the pitch, so if 10 wickets are lost, it leaves one batter stranded.

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

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 [**?**], in which they report that the table used for calculating the method can be employed by

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 matches 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." [**?**]. 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.

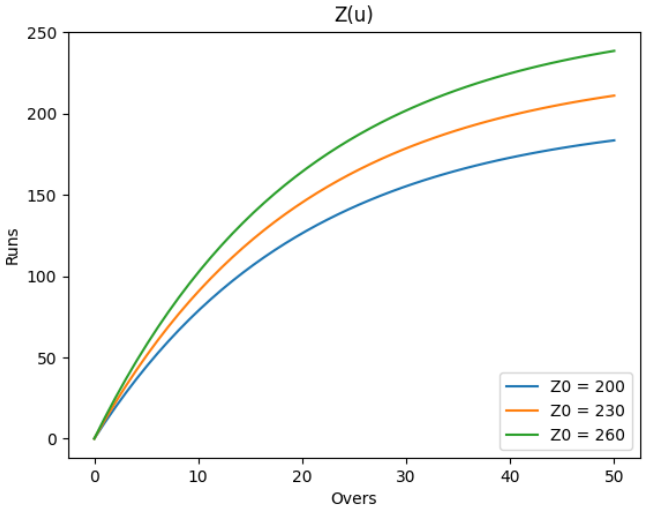
3.1 Origins: Duckworth and Lewis

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 **??** 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 **??** 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 **??** 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 **??** 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

3.3 Extension: Bayesian Modelling

This section will look at the work of paper [**?**], which used Bayesian modelling to try and improve the score predictions of the D/L method. The authors used the same dataset as we are using for the dissertation, although over a slightly smaller range of games. Only games between 2005-2017 are used, whereas our dataset goes up to 2021. Note that to keep consistent with the fact we are talking about a different model now, we swich from using $Z(u, w)$, to using $R(u, w)$, as to keep consistency with the different papers being looked at. They start by introducing the following nonlinear regression model:

$$\bar{R}(u, w) \sim N(m(u, w; \theta), \frac{\sigma^2}{n_{uw}}) \tag{3.5}$$

Where $\bar{R}(u, w)$ is the sample average of runs scored by a team from the total number of matches in the data set. We have $m(u, w; \theta)$ as the corresponding modeled population average

of runs scored by a team when a considerable amount of games are taken into account. θ denotes a vector of parameters that will be specified later on. Since $R(u, w)$ is not calculated in all games, the average score is taken over all matches where scores are present, this is given by n_{uw}. The sample mean $\bar{R}(u, w)$ is the calculated over this quantity. This is actually the point which motivates using Bayesian statistics to extend the D/L method. The authors report that approximately 26.8% of values for $R(u, w)$ were missing in the dataset, so by using a Bayesian inference model, we can use the posterior predictive distribution to account for missing values. Which in turn should increase the predictive accuracy of this model.

We return now to look at the $m(u, w; \theta)$ parameter that appears in **??**. This mean function, based on the plots produced in the original D/L paper, (see **??** for an example), the authors adopted an exponential decay. We can see why this choice makes sense from figure **??**. The resources considered by this method, namely runs and wickets, do decrease exponentially as a game progresses. $m(u, w; \theta)$ depends on the parameters $a_w = Z_0(w)$ and $b_w = b(w)$, each one depending on the number of wickets fallen at that given time, with u overs remaining. We therefore have:

$$m(u, w; \theta) = a_w(1 - exp(-b_w u)) \tag{3.6}$$

$$\theta = \{(a_w, b_w); w \in \mathcal{W} = \{0, 1, \ldots, 9\}\} \tag{3.7}$$

Before proceeding with defining the prior specifications on the parameters for this model, we need the following distribution:

Definition 3.3.1. *$X \sim Ga(a, b)$ has a **Gamma Distribution** with mean $\frac{a}{b}$ and variance $\frac{a}{b^2}$ if its probability density function is*

$$\frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}.$$

We can now define the prior specifications on the parameters. First, we fix A_0 and B_0 large enough such that $R(50, 0) < A_0$. We initialise the model with $a_0 \sim U(0, A_0)$ and $b_0 \sim U(0, B_0)$. Then given any pair (a_0, b_0) and for $w = 0, 1, \ldots, 8$, we generate:

$$a_{w+1} | \sigma^2, a_w, b_w \sim U(0, a_w) \tag{3.8}$$

$$b_{w+1} | \sigma^2, a_{w+1}, b_w, a_w \sim U\left(0, \frac{a_w b_w}{a_{w+1}}\right) \tag{3.9}$$

$$\frac{1}{\sigma^2} \sim Ga(a, b). \tag{3.10}$$

Above, U(a,b) represents a uniform distribution over the open interval (a, b). Now that the prior distribution is obtained, the likelehood function is then given by:

$$L(\theta, \sigma^2) = \left(\frac{1}{\sigma/\sqrt{n_{uw}}}\right)^{500} exp\left\{-\frac{1}{2(\sigma^2/n_{uw})} \sum_{u=1}^{50} \sum_{w=0}^{9} (R(u, w) - m(u, a_w, b_w))^2\right\} \tag{3.11}$$

The authors chose their parameters A_0, B_0, a and b such that the prior is not sensitive to the posterior inderence. The values chosen were therefore $A_0 = 200$ $B_0 = 100$ and $a = b = 0.1$.

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

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")
```