MATH5003 - Assignment in Mathematics

Modeling and Simulating Football Results

James Adam Gardner - 200318885

Supervisor - Dr Jochen Voss

May 6, 2011

Abstract

This report contains the description and implementation of a Poisson regression model being used to model football results. The steps taken to set up the model are explained in full and the statistical program R is used to carry out the numerical calculations. After modeling the football results, new games can be simulated to analyse the randomness of football league tables. Modeling and simulating the results can then be used to try to predict, or indicate the outcome of, future football matches. All of the R code used is given in the appendix and can be downloaded and used at `http://www.maths.leeds.ac.uk/~voss/projects/2010-sports/`. The data used throughout the report is taken from the website `http://www.football-data.co.uk/` [4].

Contents

Chapter 1

Introduction

The Betting Market

Trying to predict the outcome of games in sport is extremely common in the world today. Betting companies do it on a regular basis to be able set their odds appropriately so that the favourite has the lowest odds, i.e. a person placing a bet on the favourite will win less if they predict the outcome correctly as it is the most likely result. Also, the betting company has to make sure that it will make money overall. One way this could be done is by tempting gamblers into betting on the least likely outcome. Longer (higher) odds will be placed on this outcome so betters will be enticed to bet as there will be big winnings if this outcome does come true, even though it is perhaps very unlikely.

It is not only the bookies (betting companies) that try and predict what will happen. The gamblers that are going to bet on the odds that the bookies propose will also make their own predictions on the outcome. These predictions are normally made by the gambler using his/her intuition and self-knowledge about the sport they are betting on. However, there will be some who try to use certain techniques to anticipate the final results. One of these techniques is to use mathematics, mainly statistics and probability, to try and work out systematically whether a certain team or player will win an event. Odds used to be calculated by hand but are now worked out using computers so a lot maths is required.

Mathematics in Sport

Mathematics in this day and age is extremely advanced in solving the vast majority of problems, however, every sports fan knows how random sport can be, so is it possible to use mathematics to predict what will happen? If so, how often will it bring out the correct result? This report will look into a possible mathematical method that can be used to predict football results, and attempt to figure out whether this method is actually successful.

Not only will the report look at predicting football results but simulating them too to analyse the statistics of results. The analysis will include looking at the randomness of a football league

table, the difference in strengths between the teams and whether these affect the simulations and randomness and also where teams are likely to finish due to all of these. One main model will be used to try to simulate and predict results called the Poisson regression model (chapter 3). Obviously any predictions made will not be totally accurate and whilst they may be able to give a good interpretation of what is most likely to happen, it is very risky to use them for betting purposes.

All of the simulations and predictions will be carried out in the statistical program R and the source code used to do so can be found at the website `http://www.maths.leeds.ac.uk/~voss/projects/2010-sports/`. All of the functions are built in R and are explained during this report and in the appendix (A etc...) where each function is given in full and the input and output is briefly explained.

Alternatively, the following R code can be entered to download the functions straight into R (the first line of code), use the 2009-2010 data set that is used throughout the report (the second line of code) and to start with, parameters can be estimated (the third line of code).

```
source("http://www.maths.leeds.ac.uk/~voss/projects/2010-sports/Football.R")
results0910 <- read.csv("http://www.maths.leeds.ac.uk/~voss/projects/2010-sports
                        /Results0910.csv", stringsAsFactors=F, header=F)
Parameters(results0910)
```

The other functions can then be implemented using this output of parameters.

Chapter 2

Poisson Models

2.1 Poisson Process

The "events" that happen in sport are random. However, when looking at how and when these events occur during the period of a game, it can be seen that they can be viewed as a timeline of events which in turn can be interpreted as a 1-dimensional scatter diagram. Take a football match for example where the goals scored by either team throughout the 90 minutes happen at random times over the period of the 90 minute game. For a match that finishes 3-2, say, with the goals being scored in the 10th, 35th, 39th, 67th and 84th minutes (ignoring which team scored each goal), then each goal can be represented by marking the events on a 1-D scatter diagram, represented as a timeline (Figure 2.1).

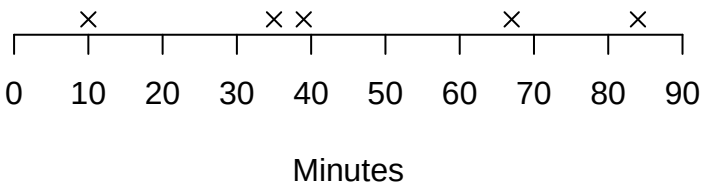

Figure 2.1: Example of times of goals in a football match which can be viewed as 1-D Poisson process

This seemingly random selection of "events", goals in the example above, can be viewed as a counting process. The crosses could be seen as representing many other events, not just goals in football. They could represent other events in sport, for example, a try in rugby, or as events not in sport, e.g. when earthquakes occur. This will depend on what you are analysing. Also,

the minutes along the bottom could be changed so that the timeline represents whichever sport, or anything else, that someone wishes to evaluate, e.g. the minutes would only go up to 80 for rugby, or for the earthquake example, they could be years over a certain time span.

2.1.1 Definition - Counting Process

For each real number $t \geq 0$, let $N(t)$ denote the number of times a specified event occurs by time t. Note that $N(t)$ is a nonnegative-integer valued random variable for each $t \geq 0$. The collection of random variables $N(t) : t \geq 0$ is called a **counting process** because it counts the number of times the specified event occurs in time. [7, page 686]

2.1.2 Definition - Poisson Process

A counting process $N(t) : t \geq 0$ is said to be a **Poisson process with rate** λ if the following three conditions hold [7, page 688]:

(a) $N(0) = 0$.

(b) $N(t) : t \geq 0$ has independent increments.

(c) $N(t) - N(s) \sim \text{Poiss}(\lambda(t - s))$ for $0 \leq s < t < \infty$.

The second condition in the definition mentions that the increments in time are independent. However, in sport, this may not be the case. Say a football match is split into 2 increments, first half and second half, then it is obvious that, whatever has happened in the first half could affect what happens in the second half. This could be for a variety of reasons such as a losing team needs a rest and then plays much better in the second half or a manager makes excellent tactical changes because his team is losing and this affects the game. This dependency could cause problems so, for the purpose of modeling football scores, it is easier to first concentrate on the game as a whole and assume independence.

2.1.3 Modelling a Poisson process

It is obvious to see that, at the start of the process where $t = 0$,

$$N(0) = 0$$

and for the example of a football match finishing 3-2 (ignoring the possibility of added time)

$$N(90) = 5.$$

It is also clear, as Weiss [7, page 686] states, that:

$$N(t) - N(s) = \text{ number of successes in the time interval } (s, t] \text{ for } 0 \leq s < t < \infty$$

This shows that even different time intervals between 0 and t can be viewed as a Poisson process.

Examples

In the example above it would be looking at goals in a football match before the 90th minute, or normally a little bit longer due to added stoppage/injury time. The process $N(t)$ is a "non-negative integer valued random variable" representing the number of occurrences of an event during time t and a collection of these random variables is what can be viewed as a Poisson process. Although for sporting games it is only possible to view the events as 1-D Poisson processes, they can be seen in higher dimensions.

Kingman [5, Page 1] gives a good example of a 2-D Poisson process by showing stars in the sky. These are positioned randomly throughout space and as they are viewed, "the stars appear to be distributed over the surface of a sphere", so if the sphere is 'flattened' out to view them as a 2-D scatter graph they can be assessed as a 2-D Poisson process, an example of which is given in Figure 2.2.

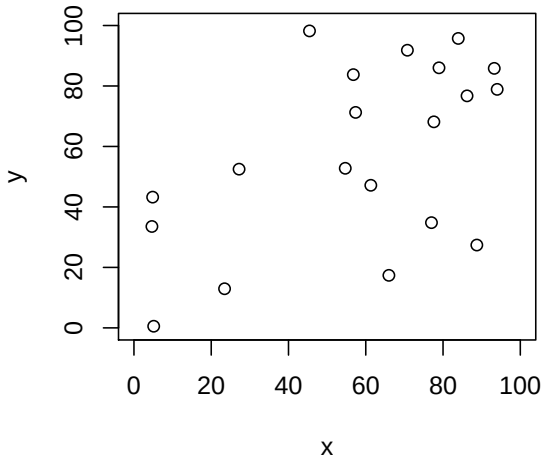

Figure 2.2: 20 points at random coordinates can be modeled as a 2-D Poisson process

The 1-D Poisson process is clearly the option that shows a football match or sport event best. This is because the events can be viewed as a timeline as explained earlier. So for the example of a football match, each match that is used as data will have a different value for $N(t)$ and the collection of these will be the Poisson model that can be tested. To analyse the probabilities, it is possible to use the similarities between the Poisson process and the Poisson distribution. The Poisson distribution will be explained later on in chapter 2.2. The link is the fact that, as Kingman [5, page 3] states, each Poisson process $N(t)$ "are not merely independent" but "that each has the Poisson distribution Poiss(μ) for a suitable choice of μ.

There is an argument here that football matches themselves are not independent due to varying strengths of the teams that will play each other. Various dependencies can affect the outcome of

a match, e.g. which team is at home, whether a team has injuries to their squad or how strong one team is compared to the other, however, for the purpose of modeling football results, it will be taken that a Poisson process $N(t)$ has a Poisson distribution. A test of how closely football results fit to a Poisson distribution will be carried out in section 2.2.4.

2.2 The Poisson Distribution

It is now clear that football scores can be viewed as a Poisson process. Therefore, the link between the Poisson process and Poisson distribution can be used. This will mean that the football scores can be modeled using the Poisson distribution.

2.2.1 Definition - Poisson Random Variable

A discrete random variable X is called a **Poisson random variable** if its probability mass function is of the form

$$p_X(x) = \exp(-\lambda)\frac{\lambda^x}{x!} \tag{2.2.1}$$

for $x = 0, 1, 2, ...$ and $p_X(x) = 0$ otherwise, where λ is some positive real number. It is said that X has the **Poisson distribution with parameter** λ. For convenience, it is sometimes written as $X \sim \text{Poiss}(\lambda)$ to indicate that X has the Poisson distribution with parameter λ. [7, page223]

The probability mass function (2.2.1) shows that the probability of the random variable X being a value x equals the right hand side of (2.2.1) and can be written as $p_X(x) = P(X = x)$. Since it is used to calculate probabilities of the number of times the event will occur, and it is clear that the total probability of the number of events to happen is equal to 1, then:

$$\sum_{x=1}^{\infty} \exp(-\lambda)\frac{\lambda^x}{x!} = 1 \tag{2.2.2}$$

for any value of λ, where λ is the expected value of the variable. A sum is used here as a Poisson process and distribution uses discrete values, whereas in the continuous case, this would be an integral as the area under the curve would be needed. However, in the discrete Poisson case, it is only required that the number of occurrences at each time is recorded as the data points, which can then be viewed in a bar chart. The graphical representation of this will look different for differing values of λ. Lower values of lambda will skew the density to the left as the expected value of the random variable is low meaning that a higher percentage of low values will occur. The bar charts (Figure 2.3) show probability mass shape for a Poisson random variable with various values of λ by processing 200 random Poisson values for each λ.

For a Poisson distribution, the value λ represents the expected number of occurrences of an event for the variable in question. A unique property of the Poisson distribution is that both the mean

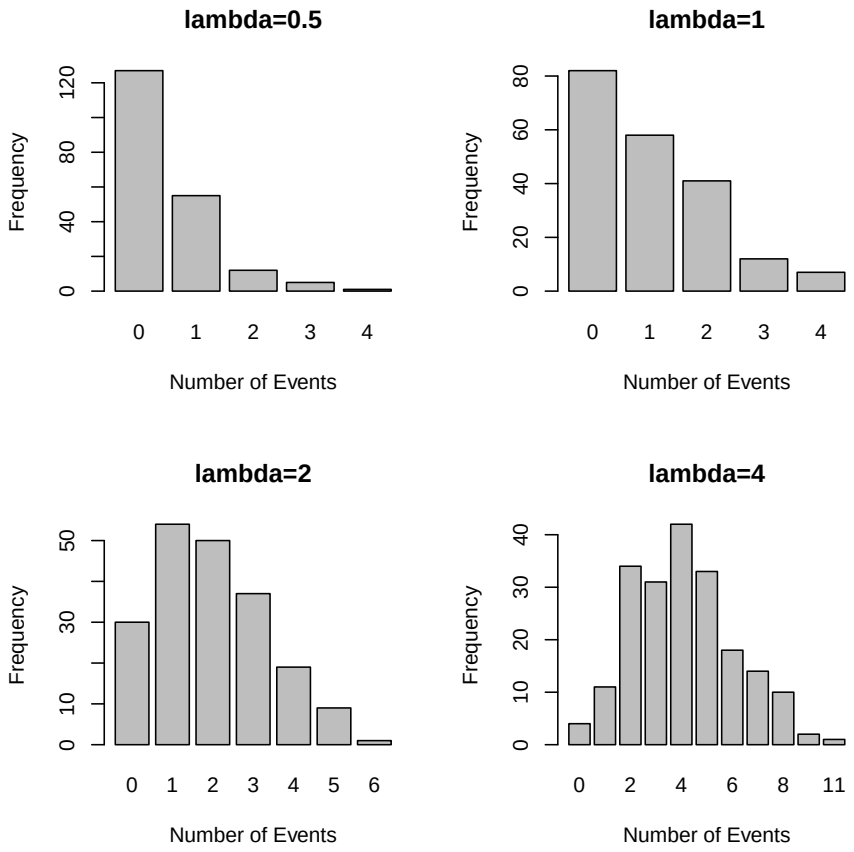

Figure 2.3: 4 Poisson variables processing 200 random values for different values of λ

the variance are equal to this value of λ.

2.2.2 Theorem - Mean and Variance of a Poisson Distribution

Let X be a discrete Poisson random variable with probability density function (2.2.2). Then $E(X) = \lambda$ and $Var(X) = \lambda$.

Proof:

Let $X \sim \text{Poiss}(\lambda)$ be a discrete Poisson random variable with probability density function (2.2.2). Take $E(X)$ first. It is known that

$$E(X) = \sum_{x=0}^{\infty} x p_X(x). \tag{2.2.3}$$

However, for the case where $x = 0$, it is clear to see that

$$E(X) = 0$$

and therefore $E(X)$ (2.2.3) can be simplified to

$$E(X) = \sum_{x=1}^{\infty} x p_X(x).$$

Now, using equation (2.2.1):

$$\begin{aligned} E(X) &= \sum_{x=1}^{\infty} x \frac{\lambda^x \exp(-\lambda)}{x!} \\ &= \sum_{x=1}^{\infty} \frac{\lambda^x \exp(-\lambda)}{(x-1)!}. \\ &= \lambda \sum_{x=1}^{\infty} \frac{\lambda^{x-1} \exp(-\lambda)}{(x-1)!}. \end{aligned}$$

Now let $u = x - 1$, $\Rightarrow$

$$E(X) = \lambda \sum_{u=0}^{\infty} \frac{\lambda^u \exp(-\lambda)}{(u)!}.$$

The equation within the sum now represents a probability mass function and therefore, by (2.2.2), the sum equals 1. Thus $E(X) = \lambda$.

Now, to show $Var(X)$, it is known that $Var(X) = E(X^2) - (E(X))^2$. So all that needs calculating now, as $E(X) = \lambda$ from above, is $E(X^2)$. This can be done in the same way as above, ignoring the $x = 0$ case again as $E(X^2) = 0$ for $x = 0$, by using the known equation:

$$\begin{aligned} E(X^2) &= \sum_{x=1}^{\infty} x^2 \frac{\lambda^x \exp(-\lambda)}{x!} \\ &= \sum_{x=1}^{\infty} x \frac{\lambda^x \exp(-\lambda)}{(x-1)!}. \end{aligned}$$

Let $u = x - 1$ again, and therefore $x = u + 1$. So,

$$\begin{aligned} E(X^2) &= \sum_{u=0}^{\infty} (u+1) \frac{\lambda^{u+1} \exp(-\lambda)}{u!} \\ &= \sum_{u=0}^{\infty} u \frac{\lambda^{u+1} \exp(-\lambda)}{u!} + \sum_{u=0}^{\infty} \frac{\lambda^{u+1} \exp(-\lambda)}{u!} \\ &= \sum_{u=0}^{\infty} \frac{\lambda^{u+1} \exp(-\lambda)}{(u-1)!} + \sum_{u=0}^{\infty} \frac{\lambda^{u+1} \exp(-\lambda)}{u!} \\ &= \lambda^2 \sum_{u=0}^{\infty} \frac{\lambda^{u-1} \exp(-\lambda)}{(u-1)!} + \lambda \sum_{u=0}^{\infty} \frac{\lambda^u \exp(-\lambda)}{u!} \\ &= \lambda \left[\lambda \sum_{u=0}^{\infty} \frac{\lambda^{u-1} \exp(-\lambda)}{(u-1)!} + \sum_{u=0}^{\infty} \frac{\lambda^u \exp(-\lambda)}{u!} \right]. \end{aligned}$$

The two equations within the summations are now both probability mass functions and therefore, by (2.2.2), both sums are equal to 1. Thus, $E(X^2) = \lambda(\lambda+1)$. Using this result and the previous result of $E(X) = \lambda$:

$$\begin{aligned} Var(X) &= E(X^2) - (E(X))^2 \\ &= \lambda(\lambda+1) - (\lambda)^2 \\ &= \lambda^2 + \lambda - \lambda^2 \\ &= \lambda. \end{aligned}$$

So $E(X) = \lambda$ and $Var(X) = \lambda$. ■

2.2.3 Modeling Football Scores as a Poisson Distribution

Taking the 2009-2010 Premiership football season as an example, where information on all 380 matches [4] is considered, and looking at the total number of goals scored by both teams in each of the 380 games played, it is easy to see that the distribution of the goals scored (Figure 2.4) looks very similar to the shape of a Poisson distribution discussed earlier. The total number of goals scored in this Premiership season was 1053, giving an average of $\frac{1053}{380} = 2.7710526...$ goals per game. Therefore, let X be the set of data of the total number of goals in all 380 matches, then $E(X) \approx 2.771053$ and from the theorem above $\lambda \approx 2.771053$. Below are two bar charts, the left hand side of Figure 2.4 is the actual data for total goals scored in each of the 380 matches from the 2009-2010 Premiership season and the other on the right hand side of Figure 2.4 is a random Poisson variable for the value $\lambda = 2.771053$ where the probabilities of each number of occurrences is calculated from $\lambda = 2.771053$ and using 2.2.1.

Comparing the two charts, it is clear to see that the goals scored in the games of the 2009-2010 Premiership season do very closely follow a Poisson distribution with a mean of the average number of goals scored in a game in that season. This can be seen due to the fact that scoring a higher number of goals than the mean, which is generally around 2 goals per game, is a lot less likely.

2.2.4 Test of Distribution

How close the total goals fits the Poisson distribution can be tested using a goodness of fit test. As the Poisson distribution and the football data are discrete, the Chi-squared goodness of fit test can be used. It tests whether observed data (the 2009-2010 Premiership football season) follows a particular distribution (the Poisson distribution). A null hypothesis of

H_0: The distribution of total goals in a game is approximately a Poisson distribution

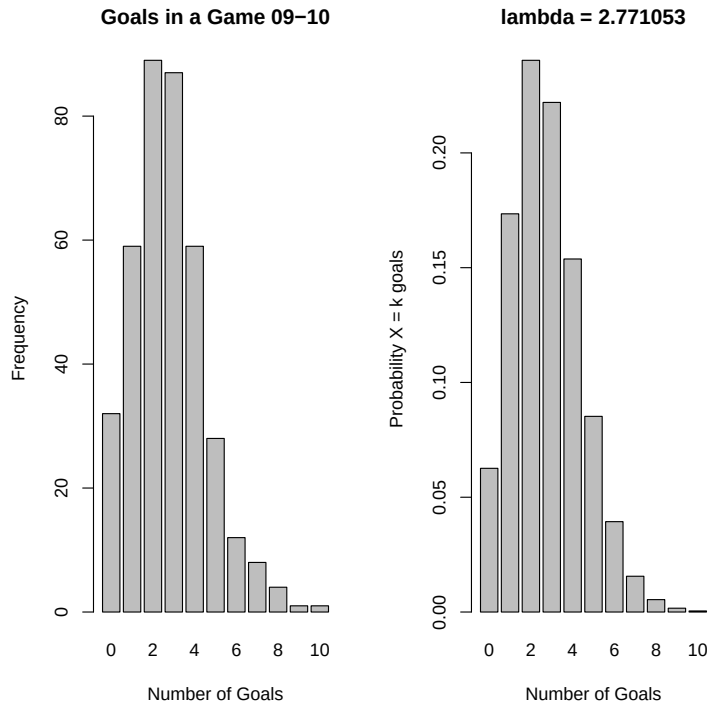

Figure 2.4: Goals scored in each of the 2009-2010 Premiership games vs Poisson probability mass function for $\lambda = 2.771053$ (X represents a football game, k represents number of goals)

is taken and the test statistic χ_t^2 is calculated using the formula

$$\chi_t^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2.2.4}$$

where O_i are the observed values from the data, E_i are the expected values for the distribution is question and n is the number of observations. The observed values from the 2009-2010 Premiership football season of total number of goals in the 380 games are the frequencies

Table 2.1: Total goals in a game in the 2009-2010 Premiership football season

Number of Goals	0	1	2	3	4	5	6	7	8	9	10
Frequency	32	59	89	87	59	28	12	8	4	1	1

which can be seen in a bar plot on the left hand side of figure 2.4. The mean number of goals scored in a game in this season was 2.771053 as discussed earlier in the chapter and the distribution to test this data against is the Poisson distribution with $\lambda = 2.771053$ whose probability mass function is displayed on the right hand side of figure 2.4. The expected frequency of total goals is calculated using the probability mass function (2.2.1) for $x = 0, ..., 10$ and $\lambda = 2.771053$. These expected values can be computed using the 'goodfit' function in the

statistical program 'R' using the R code below where the table (2.1) has been created and inputted into the goodfit function. Also the type of distribution has been specified as 'Poisson' so that the values are calculated using 2.2.1 and the method of estimation is specified as 'ML' (maximum likelihood) so that the correct method of estimation is used (this will be discussed in section 3.3).

```
> tg <- rep(0:10,c(32,59,89,87,59,28,12,8,4,1,1))
> fretabtg <- table(tg)
> fretabtg
tg
 0  1  2  3  4  5  6  7  8  9 10
32 59 89 87 59 28 12  8  4  1  1
> gf <- goodfit(fretabtg, type= "poisson", method= "ML")
> gf

Observed and fitted values for poisson distribution
with parameters estimated by 'ML'

 count observed     fitted
     0       32 23.7865102
     1       59 65.9136717
     2       89 91.3251266
     3       87 84.3555775
     4       59 58.4384363
     5       28 32.3871965
     6       12 14.9577710
     7        8  5.9212530
     8        4  2.0510130
     9        1  0.6314961
    10        1  0.1749909
```

The table gives observed values in the observed column and expected values in the fitted column as they have been fitted to the theoretical distribution being tested. Now it is possible to calculate the test statistic (2.2.4) for these values which R can provide by looking at the summary of this goodfit function.

```
> summary(gf)

         Goodness-of-fit test for poisson distribution

                      X^2 df  P(> X^2)
Likelihood Ratio 8.941242  9 0.4427157
```

This shows that $\chi_t^2 = 8.941242$ with 9 degrees of freedom and the p-value ($P(> X^2) = 0.4427157$) given means that the probability of the χ^2 value in the χ^2 table being above the value of χ_t^2

is $\approx 44\%$. Therefore it is possible to say that there is 44% chance that the distributions are similar. This is quite significant due the large amount of data and so it can be said that the null hypothesis can be accepted and the distribution of total goals from the 2009-2010 Premiership football season does in fact follow a Poisson distribution. Consequently this backs up the idea that the Poisson distribution is the correct distribution for football data and will provide a good model to use.

Chapter 3

Poisson Regression

The Poisson regression model is used as the "standard model for count data" [1, page 9] and is "derived from the Poisson distribution by allowing the intensity parameter μ to depend on covariates (regressors)". The parameter μ is represented by λ in the previous explanation of the Poisson distribution in section 2.2. The Poisson regression model follows the same steps as a normal linear regression model $y = X\beta + \epsilon$ where the scalor variables y and variables X are used to estimate the parameters β. However it is slightly different in that values need to be positive for count data which means exponentials need to be used and that the Poisson distribution is specifically used as the model. This will be explained in more detail throughout this chapter.

3.1 Definition - Count Data

An event count refers to the number of times an event occurs, for example the number of airline accidents or earthquakes. An event count is the realisation of a non-negative integer-valued random variable. [1, page 1]

3.2 Poisson Regression Model

A counting process (2.1.1) is a way of resembling sporting events or data and these events can therefore be viewed as count data [1, page 9]. The events that happen over a time period take the form of count data so that they are in a format to be analysed. This count data can in turn can be modeled using a standard nonlinear regression model known as the Poisson regression model.

The Poisson regression model comes directly from the Poisson distribution by using the same mass function and using variables which describe the occurrences of the events. In this model, y_i are the scalar dependent variables which will correspond to the number of times the event in question occurs and $\mathbf{x}_i$ is a vector of linearly independent regressors [1, page 9], i.e. a variable

that has an effect on the values y_i. Therefore, as y_i depends upon the $\mathbf{x}_i$, the model looks at the random variable $y_i|\mathbf{x}_i$, i.e. y_i given $\mathbf{x}_i$, that, following the Poisson distribution, has density

$$f(y_i|\mathbf{x}_i) = \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}. \tag{3.2.1}$$

for $i = 1, ..., n$. This is where $\mu_i = \mu(\mathbf{x}_i, \boldsymbol{\beta}) = E[y_i|\mathbf{x}_i]$ are the mean vectors with $y_i \in \mathbb{N}_0$. The dimensions of the terms matters here because otherwise the matrix-vector multiplication of $X\boldsymbol{\beta}$ in the model would fail. So $\mathbf{x}_i^T = [x_{1i}, ..., x_{di}]$ is a d-dimensional vector of the regressors x_{ji} $(j = 1, ..., d)$ where $\mathbf{x}_i \in \mathbb{R}^d$. Also, $\boldsymbol{\beta}$ gives the parameters of the model where $\boldsymbol{\beta} \in \mathbb{R}^d$ is a '$d \times 1$' sized vector. So y_i is an '$n \times 1$' vector, $\mathbf{x}_i$ is an '$n \times d$' matrix and $\boldsymbol{\beta}$ is a '$d \times 1$' vector. Therefore, matrix multiplication of $\mathbf{x}_i\boldsymbol{\beta}$ is an '$n \times 1$' which are the dimensions of y_i.

The parameters $\boldsymbol{\beta}$ are estimated using an estimation method. As Cameron and Trivedi explain [1, page21] "The standard estimator for this model is the maximum likelihood estimator". This is the case because the parameter vector $\boldsymbol{\beta}$ is unknown and "The basic idea of the maximum likelihood principle is to consider the density" [6, page33] so the Poisson density can be used. Also, the other main estimation method is Ordinary Least Squares (OLS) which requires residuals ϵ in the model. However the Poisson regression model does not contain the residuals so OLS cannot be used.

3.3 Maximum Likelihood

The parameters can be estimated using the log-linear Poisson regression model. In this model, the mean vectors are $\mu_i = \exp(\mathbf{x}_i^T\boldsymbol{\beta})$ are now the exponential mean functions for the Poisson distribution to ensure that they are positive as is required for count data. This exponential is where the "log" comes from in the name "log-linear Poisson regression model". To estimate the parameters $\boldsymbol{\beta}$, the method of maximum likelihood estimation is used as just discussed at the end of section 3.2. The maximum likelihood estimator (MLE) of the parameter vector is calculated as a function of the parameter vector [6, page33]. Therefore this estimation is computed using the likelihood function (3.3.1) which takes a product of all the densities (3.2.1) from $i = 1, ...n$ and takes the logarithm of this product due to the exponential in the mean vector.

$$\mathcal{L}(\boldsymbol{\beta}; y_i, \mathbf{x}_i) = \log\left(\prod_{i=1}^{n} f(y_i|\mathbf{x}_i)\right) \tag{3.3.1}$$

Therefore using (3.2.1) and that $\mu_i = \exp(\mathbf{x}_i^T\boldsymbol{\beta})$, the likelihood function becomes:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta}; y_i, \mathbf{x}_i) &= \log\left(\prod_{i=1}^{n} \frac{\exp(-\mu_i)\mu_i^{y_i}}{y_i!}\right) \\
&= \log\left(\prod_{i=1}^{n} \frac{(\exp(\mathbf{x}_i^T\boldsymbol{\beta}))^{y_i} \exp(-\exp(\mathbf{x}_i^T\boldsymbol{\beta}))}{y_i!}\right) \\
&= \sum_{i=1}^{n} \log\left(\frac{\exp(y_i\mathbf{x}_i^T\boldsymbol{\beta}) \exp(-\exp(\mathbf{x}_i^T\boldsymbol{\beta}))}{y_i!}\right) \\
&= \sum_{i=1}^{n} \left(y_i\mathbf{x}_i^T\boldsymbol{\beta} - \exp(\mathbf{x}_i^T\boldsymbol{\beta}) - \log(y_i!)\right).
\end{aligned}$$

The last line of these equations is given by Cameron and Trivedi [1, page 21] (equation 2.5) but the steps taken to get from the likelihood function (3.3.1) are not given in the text. The next step is to estimate the parameter vector and get to 3.3.2 (equation 2.6 in the text) but they do not give the full working here either.

So, to estimate the parameter vector $\boldsymbol{\beta}$ as required, first differentiate the likelihood function with respect to each individual parameter β_j where $j = 1, ..., d$.

$$\begin{aligned}
\frac{\delta}{\delta\beta_j}\mathcal{L}(\boldsymbol{\beta}) &= \sum_{i=1}^{n} y_i x_{ij} - x_{ij} \exp(\mathbf{x}_i^T\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} \left(y_i - \exp(\mathbf{x}_i^T\boldsymbol{\beta})\right) x_{ij}
\end{aligned}$$

Then setting this derivative equal to 0, $\hat{\boldsymbol{\beta}}$ replaces $\boldsymbol{\beta}$ because it is now an estimate for the parameter vector. Therefore the equation below is achieved as Cameron and Trivedi give.

$$\sum_{i=1}^{n} \left(y_i - \exp(\mathbf{x}_i^T\hat{\boldsymbol{\beta}})\right) x_{ij} = 0 \tag{3.3.2}$$

As stated [1, page 21], there is no analytical solution for $\hat{\boldsymbol{\beta}}$ from equation 3.3.2 and it would have to be calculated numerically. By hand, this is very difficult and would take too long to compute. However, these numerical calculations can be carried out by a computer and the statistical program R is a good environment to take care of this. The function in R that estimates the parameters is called `glm` standing for 'generalised linear model'. For the `glm` function, the inputs needed are a formula and the family of model that is to be used. So the formula to input for the Poisson regression model is

$$Y = \exp(X\boldsymbol{\beta}) \tag{3.3.3}$$

where $Y = Y_i^T$ is the vector of data, $X = \mathbf{x}_i^T$ represents the matrix for the regressors and $\boldsymbol{\beta}$ is the vector of parameters. The actual formula inputted into the `glm` function will be $Y = X\boldsymbol{\beta}$ and specifying the family of the model to be Poisson will initiate the exponential. A simple example of the function set up is

```
glm(formula = Y ~ X, family = poisson)
```

where `glm` will add $\boldsymbol{\beta}$ to Y and X and calculate it from equation 3.3.3 numerically. However, the matrix X must have full rank for the parameter vector $\boldsymbol{\beta}$ to be estimated using maximum likelihood, i.e. $\text{rank}(X) = d$ as there are d parameters to be estimated. This is pointed out by Toutenburg [6, page23-24] in 'Best Linear Unbiased Estimation' where to estimate to the unknown parameter vector $\boldsymbol{\beta}$, "X is of order $T \times K$, with full rank K". So for the set up above, $T = n$ and $K = d$.

3.4 Implementing Poisson Regression in R

Now the process of Poisson regression has been explained, the steps outlined above can be implemented in the statistical program R via the `glm` function. To create the model in R the two variables y_i and $\mathbf{x}_i$ will be used, which are the scalar dependent variables and linearly independent regressors respectively. From section 3.2, it is known that $\mu_i = E[y_i|\mathbf{x}_i]$ are the mean vectors and that $\mu_i = \exp(\mathbf{x}_i^T\boldsymbol{\beta})$ where $\boldsymbol{\beta}$ are the parameters of the model. Therefore, as a Poisson process has the property $E[X] = \lambda$, the Poisson model in R will be based on

$$\lambda = \exp(\mathbf{x}_i^T\boldsymbol{\beta}). \tag{3.4.1}$$

For a Poisson distribution, $Y \sim \text{Poiss}(\lambda)$ where Y is the data is question. So plugging in (3.4.1) this then becomes

$$Y_i \sim \text{Poiss}(\exp(\mathbf{x}_i^T\boldsymbol{\beta})) \tag{3.4.2}$$

where Y_i is a vector, $\mathbf{x}_i^T$ is a matrix and $\boldsymbol{\beta}$ is a vector. The elements of these vectors and matrices that will be used will be explained later in chapter 4.2 where football data will be applied to the method described during this chapter.

Chapter 4

Poisson Regression and Football Data

The next stage is to apply the Poisson Regression Model to real football data. Full data from the top 4 leagues in England can be found on the Internet [4] along with data for many other leagues across Europe. The data for a season of a particular league from a particular season (the earliest available dating back to the 1993-1994 season for a lot of the leagues) comes in the form of an Excel spreadsheet where numerous amounts of statistics are given for each game in that season. For the model, the only 4 bits of data from each game that will be used for now are 'home team', 'away team', 'home goals' and 'away goals'. So by downloading the spreadsheet wanted and deleting all but the 4 columns of data needed and saving the file as a 'comma separated value' spreadsheet, the season's results are ready to input into R.

4.1 Football Data

Take the data set used earlier of the 2009-2010 Premiership football season which, as described above, is in the form of a 'comma separated value' spreadsheet. This contains the 4 bits of data for each of the 380 games in this season that are required. This spreadsheet can be imported into the statistical program R using the command below.

```
> results0910 <- read.table("Results0910.csv", sep=",", stringsAsFactors=F)
> results0910
             V1          V2 V3 V4
1   Aston Villa       Wigan  0  2
2     Blackburn    Man City  0  2
3        Bolton  Sunderland  0  1
...
378  Man United       Stoke  4  0
379    West Ham    Man City  1  1
380      Wolves  Sunderland  2  1
```

So for each of the 20 teams in the Premiership in the 2009-2010 season, there are home and away results against each of the other 19 teams. The table gives the home team in column 1, away team in column 2, home team score in column 3 and away team score in column 4. There are 380 games because there were 20 teams in the Premiership in the 2009-2010 season, as there has been since 1995 when the number of teams was reduced from 22 to 20, and each of the 20 teams has to play each of the other 19 teams twice (one at home and one away). So the first team will play 19 home games against the other 19 teams, as will all of the other 19 teams. This therefore includes each team's away fixture against each of the other 19 teams as well. Therefore there are $20 \times 19 = 380$ games which matches the table `results0910` as there are 380 rows of data depicting the 380 games of the season.

Now all of the results are inputted into R, there needs to be a way to view the league table of these games. This could just be found on the Internet, however for later on, a way to create a table will be required. To create this table, a function can be constructed in R where an empty data frame is created to start with. The row names of the data frame are the names of the 20 teams and the column names are the shortened names of the statistics that are normally on view in a football table. These headings are displayed below in table 4.1.

Table 4.1: Table heading abbreviations

Abbreviation	Meaning
P	Number of games played
HW	Number of home games won
HD	Number of home games drawn
HL	Number of home games lost
HF	Number of goals scored at home
HA	Number of goals conceded at home
AW	Number of away games won
AD	Number of away games drawn
AL	Number of away games lost
AF	Number of goals scored away
AA	Number of goals conceded away
GD	Goal difference, total goals scored - total goals conceded
Points	Number of points achieved

Then by inserting a loop into the function, it can look at each of the 380 games and add the correct numbers into the correct places in the data frame. So for each game, numerous 'if' statements are used within the function to ascertain which of two teams has scored the most goals, or whether it was a draw. These 'if' functions will ascertain where to add numbers to.

For example, the first game in the results table is `Aston Villa 0-2 Wigan` so the 'if' statement asking if the away team's goals is larger will take effect. In this case the function would add '1' to each of the teams' 'P' column for games played. Then a '1' would be added to Aston Villa's 'HL' column (a home loss) and to Wigan's 'AW' column (an away win). Next, '0' would be

added to Aston Villa's 'HF' column (0 goals scored at home) and to Wigan's 'AA' column (0 goals conceded away) and '2' would be added to Aston Villa's 'HA' column (2 goals conceded at home) and to Wigan's 'AF' column (2 goals scored away). Then '-2' would be added to Aston Villa's 'GD' column (2 more goals conceded than scored) and '2' would be added to Wigan's 'GD' column (2 more goals scored than conceded). Finally, '3' would be added to Wigan's 'Points' column (3 points for the win) and '0' would be added to Aston Villa's 'Points' column (0 points for the loss).

In modern football, this is the common format where 3 points are awarded for a win, 1 point for a draw and 0 points for a loss. This was brought in in 1981 into the English set up and has had a big effect on the game in that teams are forced to try and win games more than they used to because there are more points on offer.

The function, named `Table`, is shown in full in appendix A. This creates a table which is sorted so that the teams, and their stats from the set of games, are given in order of points gained. However, in the case where the number of points gained by two teams are equal, then the teams are sorted by who has the best goal difference. The input to the function must be a set of games in the same format as shown above (`results0910`) for the results from the 2009-2010 season. These results are labeled as `results0910` and can be inputted into this table function and the resulting table labeled as `Table0910`, both of which are shown below.

```
> Table0910 <- Table(results0910)
> Table0910
          Team  P HW HD HL HF HA AW AD AL AF AA  GD Points
1      Chelsea 38 17  1  1 68 14 10  4  5 35 18  71     86
2   Man United 38 16  1  2 52 12 11  3  5 34 16  58     85
3      Arsenal 38 15  2  2 48 15  8  4  7 35 26  42     75
4    Tottenham 38 14  2  3 40 12  7  5  7 27 29  26     70
5     Man City 38 12  4  3 41 20  6  9  4 32 25  28     67
6  Aston Villa 38  8  8  3 29 16  9  5  5 23 23  13     64
7    Liverpool 38 13  3  3 43 15  5  6  8 18 20  26     63
8      Everton 38 11  6  2 35 21  5  7  7 25 28  11     61
9   Birmingham 38  8  9  2 19 13  5  2 12 19 34  -9     50
10   Blackburn 38 10  6  3 28 18  3  5 11 13 37 -14     50
11       Stoke 38  7  6  6 24 21  4  8  7 10 27 -14     47
12      Fulham 38 11  3  5 27 15  1  7 11 12 31  -7     46
13  Sunderland 38  9  7  3 32 19  2  4 13 16 37  -8     44
14      Bolton 38  6  6  7 26 31  4  3 12 16 36 -25     39
15      Wolves 38  5  6  8 13 22  4  5 10 19 34 -24     38
16       Wigan 38  6  7  6 19 24  3  2 14 18 55 -42     36
17    West Ham 38  7  5  7 30 29  1  6 12 17 37 -19     35
18     Burnley 38  7  5  7 25 30  1  1 17 17 52 -40     30
19        Hull 38  6  6  7 22 29  0  6 13 12 46 -41     30
20  Portsmouth 38  5  3 11 24 32  2  4 13 10 34 -32     28
```

This set of results is for the complete 380 games from the 2009-2010 Premiership football season and therefore the table shows the final standings of the teams [1]. As can be seen, Chelsea won the title and Burnley, Hull and Portsmouth were relegated to the lower division, as is the case for every Premiership season that the bottom 3 are relegated.

The function to create the table can be implemented for any number of games that are inputted meaning that a mid-season table can also be made. Also, if the set of games has more or less teams in it then the table function is able to count the number of teams and make the table the correct size, i.e. the Championship has 24 teams so the table would be larger by 4 rows.

4.2 Poisson Regression for Football Data

The next step of this procedure is to input the football data into the Poisson regression model. The football data had been inputted into R, now it needs to be explained how to fit the football data to the Poisson regression model and then to implement the model into R with the inputted football data.

For football data, let n be the number of teams and g be the number of games being used as the data. Then the model would use $i = 1, ..., 2g$ as the data for the number of goals scored by the home teams and away teams in each of the g games, hence i goes all the way up to $2g$. It would also use $d = 2n$ because sensible parameters for the model to have would be home and away parameters for each team. These parameters would represent strengths corresponding to how good each team is at home and away compared to the rest of the teams in the league. This could change to $d = 2n + 1$ where a home advantage parameter is included. This additional parameter will be discussed later on in section 4.3 as to whether its inclusion is necessary.

From the formula 3.4.2, Y is the data and will be the scores of the teams involved in the football matches. As Y_i is a vector, it can be constructed by inserting all of the teams scores in order to create it. So if the data contains g amount of games then the length of Y_i will be $2g$ as there will be two bits of data for each game, the home team's score and the away team's score. It is clear that $Y_i \in \mathbb{N}_0$ as the team's scores will be an integer of 0 or larger, with large numbers being very unlikely, and $i = 1, ..., 2g$. So the vector will be of the form

$$Y^T = (y^1_{i_1,j_1}, y^1_{j_1,i_1}, y^2_{i_2,j_2}, y^2_{j_2,i_2}, ..., y^g_{i_g,j_g}, y^g_{j_g,i_g})$$

where $y^c_{a,b}$ represents the number of goals scored by team a versus team b in game c. For the purpose of the Poisson regression model, these scores are assumed to be Poisson distributed which is explained in section 2.2.3. Therefore the scores in Y are assumed to have a mean λ shown in 3.4.1. The mean is therefore influenced by what games took place, i.e. where the

[1] This table does not quite represent the actual 2009-2010 Premiership final standings. Portsmouth actually only gained 19 points after being deducted 9 points due to going into administration. However, fortunately, this does not change the look of the table calculated by R above so it will have little or no effect.

numbers are entered in $\mathbf{x}_i^T$ to give the correct team parameters for each game which in turn give λ for each individual score.

The $\boldsymbol{\beta}$ is a vector containing the parameters for the model, as described in chapter 3.4. In this case, $\boldsymbol{\beta}$ is of length $2n$ because each of the n teams will have an attack and defence strength. So the first half of the vector will contain the attack strengths, α_k say, of the n teams and second half will contain the defence strengths, γ_k say, of the n teams. So the vector will be of the form

$$\boldsymbol{\beta}^T = (\alpha_1, ..., \alpha_n, \gamma_1, ..., \gamma_n). \tag{4.2.1}$$

The linearly independent regressors $\mathbf{x}_i^T \in \mathbb{R}^d$ are vectors collected together to form a matrix where $i = 1, ..., 2g$, as is the case for Y_i, because there needs to be a vector for each of the two team's scores in each game and $d = 2n$ where n is the number of different teams. This $\mathbf{x}_i^T$ will be constructed so that it picks out the correct parameters to work out a particular team's score. The transpose forces the vector for each score to be the rows of the matrix so that when multiplied by the parameters (a column vector) the calculation will be correct due to the fact that a matrix times a vector is done by multiplying the rows by the vector. Each vector is of length $2n$ because the first half (n elements) will pick out the attack strength of the team's score it is calculating and the second half (n elements) will pick out the defence strength of the team they are playing. To do this for a game between team a and team b say, a number '1' will be placed in the a^{th} place of the first n elements of $\mathbf{x}_i$ (with the rest being 0's to ignore the other attack strengths) and then a '-1' will be placed in the b^{th} place of the second n elements of $\mathbf{x}_i$ (again the rest being 0's to ignore the other defence strengths). So a vector for one team in a game would be of the form

$$\mathbf{x}^T = (0, ..., 1, ..., 0, 0, ..., -1, ..., 0). \tag{4.2.2}$$

So to check these 2 vectors and matrix make sense, check the dimensions of the multiplication add up correctly. For simplicity, let the equation be as described in 3.3.3 where $Y = Y_i^T$ is the vector of data of the goals scored, $X = \mathbf{x}_i^T$ represents the matrix for all the games and $\boldsymbol{\beta}$ is the vector of parameters. X is of size $2g \times 2n$ and $\boldsymbol{\beta}$ is of size $2n \times 1$ and therefore, when multiplied together, give a vector of size $2g \times 1$ which are the dimensions of Y. This implies that the calculation will work and that the vector Y will take the form

$$Y = (\exp(\alpha_a - \gamma_b), \exp(\alpha_b - \gamma_a), ...) \tag{4.2.3}$$

where team a plays team b. However as can be seen here, $\exp(\alpha_a - \gamma_b)$ is the λ discussed earlier and therefore the values given in Y are actually just values of λ for the goals scored for each team. Therefore, to predict how many goals a team will score, these λs will be used as the mean of different Poisson distributions. How this works will be described later in section 5.

Example

Take a small example of 3 football matches finishing as follows

Table 4.2: 3 example football results

Game	Home	Away	Home goals	Away goals
1	Arsenal	Chelsea	3	2
2	Liverpool	Arsenal	1	4
3	Chelsea	Liverpool	5	0

then the vectors Y and β and matrix X described above would be represented as shown below.

$$Y^T = (3, 2, 1, 4, 5, 0) \tag{4.2.4}$$

$$\beta^T = (\alpha_{Ars}, \alpha_{Che}, \alpha_{Liv}, \gamma_{Ars}, \gamma_{Che}, \gamma_{Liv}) \tag{4.2.5}$$

$$\mathbf{X} = \begin{array}{c} \\ game1 \\ game1 \\ game2 \\ game2 \\ game3 \\ game3 \end{array} \begin{array}{c} \begin{array}{cccccc} Ars & Che & Liv & Ars & Che & Liv \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & 0 \end{pmatrix} \end{array} \tag{4.2.6}$$

Therefore, as described in the general case in equation (4.2.3), the multiplication of $X\beta$ for equation 3.3.3 gives

$$(X\beta)^T = (\alpha_{Ars} - \gamma_{Che}, \alpha_{Che} - \gamma_{Ars}, \alpha_{Liv} - \gamma_{Ars}, \alpha_{Ars} - \gamma_{Liv}, \alpha_{Che} - \gamma_{Liv}, \alpha_{Liv} - \gamma_{Che}). \tag{4.2.7}$$

So from equation 3.3.3, exponentials of each of these elements will be taken to give the vector Y. The elements of Y represent values of λ for the number of goals to be scored by a team in a particular game. How these are to be used will be discussed in section 5.1.

4.3 The Home Advantage Parameter

The setup in the previous section is good, however, there is no distinction between the team at home and the team away. Therefore an extra parameter can be added to give the home team an advantage. A home advantage parameter would give the home team an extra weighting because, in football, it is generally the case that whoever is playing at home in a game has an advantage.

This could be for a variety of reasons such as having large amounts of their own fans there or they are used to playing at their home ground. A similar model for estimating parameters is used by Davison ([3, page498]) and he also used a home advantage parameter which gives an idea that it will prove significant.

A test can be done to see if there is a need for this parameter. An appropriate test is a significance test. Take the two sample t-test which can test the difference between two means. To find out whether there should be a home advantage parameter, this test can be used on the two samples from the football data of home goals and away goals. Testing the difference of the means of these two samples will show whether there is a clear difference between them, i.e. do teams score more goals at home than away.

An example of this type of test can be found in "A basic course in Statistics" [2, page416] by Clarke and Cooke. There are 380 games in the football data from the 2009-2010 Premiership season so each of the samples, X_1 (home) and X_2 (away), have 380 bits of data which means that a large sample test can be used. The t-test requires the samples to be normally distributed but the scores, as discussed in section 2.2.3, are in fact Poisson distributed. The Poisson distribution is a skewed distribution but, as Clarke and Cooke state, "If X has a very skew distribution, we need a sample of considerable size (say, at least 200) before the distribution of $\bar{X}$ approaches reasonably closely to a normal distribution" [2, page416]. Therefore, as both samples contain 380 bits of data, a large two-sample t-test can be carried out on the data. Example 17.8 given by Clarke and Cooke [2, page417] is a perfect test to follow as it is a "Large-sample test on difference of means".

4.3.1 Test for Home Advantage

The model for the test will use the two samples from the 2009-2010 Premiership football season of home and away goals. The set of home goals scored will be represented as X_1 and the away goals as X_2 where $X_1 \sim \mathrm{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathrm{N}(\mu_2, \sigma_2^2)$.

To test whether the two means are different, let the null hypothesis (H_0) and alternative hypothesis (H_1) to be as shown below.

$$H_0 : \mu_1 = \mu_2$$

$$H_1 : \mu_1 \neq \mu_2$$

As the test is whether the means are different and not if one is larger than the other, this is a two-sided test. Under H_0, $\mu_0 = 0$ where $\mu_0 = \mu_1 - \mu_2$ and

$$\bar{x_1} - \bar{x_2} \sim \mathrm{N}\left(0, \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)$$

where n_1 and n_2 are the number of elements of data in each sample. To carry out the test, the figures of $\bar{x_1}$ and $\bar{x_2}$ are needed, as well as estimates s_1^2 and s_2^2 (sample variances from the data) of σ_1^2 and σ_2^2 respectively. These are calculated below in R from the data.

```
> mean(results0910[,3])
[1] 1.697368
> mean(results0910[,4])
[1] 1.073684
> var(results0910[,3])
[1] 2.153555
> var(results0910[,4])
[1] 1.261047
```

Now these values are known, the test statistic for the t-test can be computed using the formula below.

$$Z = \frac{(\bar{x_1} - \bar{x_2}) - \mu_0}{\sqrt{\sigma^2}}$$

This is where the estimates of s_1^2 and s_2^2 are used to compute σ^2 as follows.

$$\begin{aligned} \sigma^2 &= Var(\bar{x_1} - \bar{x_2}) \\ &= \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} \\ &= \frac{2.153555}{380} + \frac{1.261047}{380} \\ &= 0.00898579... \end{aligned}$$

Therefore, under H_0, the test statistic is as below.

$$\begin{aligned} Z &= \frac{(1.697368 - 1.073684) - 0}{\sqrt{0.00898579...}} \\ &= 6.5794... \end{aligned}$$

To see if this value is significant or not, the student's t-distribution table gives p-values at various significance levels. For the standard normal variable, the values from the ∞ row of the table are used. So for the 1% significance level, the p-value is $P(1\%) = 2.576$ and therefore, as $Z > P(1\%)$, H_0 can be strongly rejected at the 99% confidence level. Also, $Z > P(0.1\%) = 3.291$ which means that, at a 99.9% confidence level, it can be said that then means of home goals and away goals from the 2009-2010 seasons data are different. It is clear that it is the home goals that are the biggest as the mean is $\bar{x_1} \approx 1.7$ and the mean for away goals is only $\bar{x_2} \approx 1.1$.

This very significant test indicates that the home advantage parameter is clearly needed so that the model represents the data as close as possible. The necessary inclusion of the home parameter could also be seen using common sense. If it was not taken then the order of the data for each game wouldn't matter. Take a game inputted into R in the form

```
Team1     Team2     Score1     Score2
```

as one of the rows/games entered (an example can be seen in `results0910` from section 4.1). As before, `Team1` corresponds to the home team, `Team2` to the away team, `Score1` to the home team's score and `Score2` to the away team's score. Therefore, without the home advantage parameter included, this game could be inputted as

```
Team2     Team1     Score2     Score1
```

and when the parameters are estimated, the change would make no difference. So if a team is better at home in the real data then this would not follow through in the simulated environment e.g. a team's poor away record would even out the good home record. When a table is then simulated, every team's home and away records would be very similar which is not the case in real life (see `Table0910` where clearly most teams are better at home). This needs to translate into the simulation and hence the use of a home advantage parameter so the simulated parameters will represent the real data much more accurately.

This home advantage parameter, call it δ, will be added to the parameter vector $\boldsymbol{\beta}$ as it is an extra parameter that needs estimating from the data. Therefore 4.2.1 becomes

$$\boldsymbol{\beta}^T = (\alpha_1, ..., \alpha_n, \gamma_1, ..., \gamma_n, \delta). \tag{4.3.1}$$

and now has dimension $(2n + 1) \times 1$. Also, equation 4.2.2 changes. Before it represented any team in a game but now there will be a difference between a home and away rows. The row for a home team will now be

$$\mathbf{x}^T = (0, ..., 1, ..., 0, 0, ..., -1, ..., 0, 1) \tag{4.3.2}$$

where it is formulated as before but with a '1' at the end to pick out the home parameter from $\boldsymbol{\beta}$. The away row will be the same but with a '0' at the end instead as to ignore the home parameter for the away team. So it will be

$$\mathbf{x}^T = (0, ..., 1, ..., 0, 0, ..., -1, ..., 0, 0). \tag{4.3.3}$$

So each $\mathbf{x}_i$ is now of dimension $(2n + 1) \times 1$ meaning that X now has dimension $2g \times (2n + 1)$. These dimensions still work because when the multiplication in 3.3.3 is carried out, $X\boldsymbol{\beta}$ has

dimension $2g \times 1$ which is the dimension of Y. The new Y, as given previously in equation 4.2.3, is

$$Y = (\exp(\alpha_a - \gamma_b), \exp(\alpha_b - \gamma_a), ...) \tag{4.3.4}$$

where team a (at home) plays team b.

Example

Referring back to the example in section 4.2 and adding in this home parameter, the equations change slightly. The equation for Y (4.2.4) remains the same but the equations for $\mathbf{X}$ (4.2.6) and $\boldsymbol{\beta}$ (4.2.5) do change to become

$$\beta^T = (\alpha_{Ars}, \alpha_{Che}, \alpha_{Liv}, \gamma_{Ars}, \gamma_{Che}, \gamma_{Liv}, \delta) \tag{4.3.5}$$

$$\mathbf{X} = \begin{array}{c} \\ game1 \\ game1 \\ game2 \\ game2 \\ game3 \\ game3 \end{array} \begin{array}{c} \begin{array}{ccccccc} Ars & Che & Liv & Ars & Che & Liv & Home \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \end{pmatrix} \end{array}. \tag{4.3.6}$$

Also, these changes mean that equation 4.2.7 becomes

$$(X\beta)^T = (\alpha_{Ars} - \gamma_{Che} + \delta, \alpha_{Che} - \gamma_{Ars}, \alpha_{Liv} - \gamma_{Ars} + \delta, \alpha_{Ars} - \gamma_{Liv}, \alpha_{Che} - \gamma_{Liv} + \delta, \alpha_{Liv} - \gamma_{Che}). \tag{4.3.7}$$

4.4 Formulating Y and X in R

So now it is known how to input football data into R and how the data need to be set up for the generalised linear model (Poisson regression model). Now this needs to be implemented.

Firstly, to create Y, which is the vector of goals scored in the games, the following R code is used.

```
g <- nrow(games)
Y <- matrix(0,2*g,1)
for (i in 1:g) {
    Y[((2*i)-1)] <- games[i,3]
    Y[(2*i)] <- games[i,4]
}
```

This requires a table of `games` in the same format as `results0910` from section 4.1. The number of games in this table is represented by the number of rows and this number of games is assigned to the letter `g`. Then a vector named Y is created but full of 0's ready to be comprised of the scores. The size of Y is $2g \times 1$ where g is the number of games as in the description in section 4.2. The `for` loop then goes through each of the games and places the two scores of the teams involved in each game into Y. So the first game's scores will go into places 1 and 2 of Y, the second game's scores into places 3 and 4 and so on.

Now the matrix X needs to be created. The R code below builds the matrix so that each game in the data (`games`) is represented by two rows as in equations 4.3.2 and 4.3.3.

```
teams <- sort(unique(c(games[,1], games[,2])), decreasing = FALSE)
n <- length(teams)
X <- matrix(0,2*g,((2*n)+1))
for (i in 1:g) {
    M <- which(teams == games[i,1])
    N <- which(teams == games[i,2])
    X[((2*i)-1),M] <- 1
    X[((2*i)-1),N+n] <- -1
    X[(2*i),N] <- 1
    X[(2*i),M+n] <- -1
    X[((2*i)-1),((2*n)+1)] <- 1
}
```

For the same input of `games`, an alphabetically ordered vector of team names is generated from the two columns of home and away team names by picking out all of the different names. Duplicates are avoided using the `unique` command. The number of teams n is then labeled as the length of this vector. Matrix X is then created of dimension $2g \times (2n + 1)$ as is now the case with the home advantage parameter and the matrix is filled with 0s. This works well because most of the matrix will be entries of 0 with either 2 or 3 entries in each row depending on whether it is a home or away row. These entries are then entered using the for loop in the function. The `which` command asks which number element a team name from `games` is in the vector `teams` and produces this number.

Before the model is implemented in R using the `glm` function described in section 3.3, the rank of the matrix X needs to be checked. Using the R function `qr` which gives a selection of stats on a matrix, including it's rank which can be extracted on its own as shown below. This is calculated using the data `results0910` from section 4.1 and building the matrix X using the code above.

```
> x <- qr(X)
> x$rank
[1] 40
```

So the rank of the matrix X is 40, which is equal to $2n$ where n is the number of teams in the 2009-2010 Premiership season in which 20 teams made up the league. Therefore the matrix does

not have full rank of $2n+1$ which is required for estimation of the parameter vector as explained at the end of section 3.3. As the rank is less than the full value, as Toutenburg explains [6, page26], the parameter vector cannot be estimated.

This implies that one of the parameters is redundant and the matrix X needs to be shortened by a column. Taking the first column away from X

```
XX <- X[,-1]
```

will then give it dimensions of $2g \times 2n$ and therefore the parameter vector $\boldsymbol{\beta}$ now has to have dimensions $2n \times 1$ for the multiplication of the two to give the correct dimensions for Y of $2g \times 1$. So the redundant parameter (used as the first parameter for ease) has been removed from the model. This parameter is fixed arbitrarily to be 0, so let the first parameter $\alpha_1 = 0$ so that in R it is easy to remove. Therefore, as Arsenal are the first team alphabetically in the 2009-2010 Premiership season, Arsenal's attack strength is being set to 0. Davison ([3, page499]) also used this technique of setting one parameter to be 0, which happens to be Arsenal's as well, probably due to them being first alphabetically. This actually makes sense because now the other 40 parameters (19 attack strengths, 20 defence strengths and the home advantage) can be calculated by using this as the benchmark. Since Arsenal are one of the best teams in the league, it can be expected that only Manchester United and Chelsea (who were the only two teams to come above Arsenal in the league in 2009-2010, see `Table0910` in section 4.1) will have strengths above 0 and the majority of other strengths will be below 0.

4.4.1 Estimating the Parameter Vector $\boldsymbol{\beta}$

Using data inputted into R, it is possible to estimate the parameters using a function called `glm` which stands for 'generalised linear model'. The Poisson regression model is a form of generalised linear model but with the Poisson distribution taken into account, i.e. when estimating the parameters using maximum likelihood, the Poisson mass function is used as the formulas show in section 3.3. The glm function is described earlier in section 3.3 and estimates the 40 remaining parameters, using the benchmark of the removed parameter, after this redundant parameter is removed. Vector Y and matrix X are inputted into the function which the code for is shown below, where XX is X with one column (the first) removed and the 0 is added to represent the arbitrary first parameter.

```
glm(formula = Y ~ 0 + XX, family = poisson)
```

This sets up the formula $Y = \exp(X\boldsymbol{\beta})$ with the new X with one column removed and the first parameter fixed at 0. The family of the generalised linear model is specified as Poisson and then the function will carry out the numerical estimation of the parameter vector after using the maximum likelihood estimation (3.3).

A list of 40 parameters is outputted from this function and then the arbitrary parameter can be added to the front of these. So, instead of implementing each of these steps separately, i.e.

creating Y, X and using `glm`, the function `Parameters` (appendix B) can be used. This has an input of a set of games in the same style as `results0910` and will produce the 41 parameters and create a neat output of them in one step. The set of results already used in all of the earlier examples is inputted below into the function to give the output shown. Here, from the earlier description, the α_i are the attack strengths of the teams, the γ_i are the defence strengths of the teams and δ is the home advantage.

```
> TeamParameters <- Parameters(results0910)
> TeamParameters
$teams
                 Attack     Defence
Arsenal      0.00000000 -0.29966558
Aston Villa -0.47087219 -0.21825368
Birmingham  -0.77721659 -0.39136711
Blackburn   -0.69316106 -0.55183181
Bolton      -0.65696644 -0.75069397
Burnley     -0.64168730 -0.95337217
Chelsea      0.20736792 -0.07159737
Everton     -0.31741611 -0.45507449
Fulham      -0.75218030 -0.37080784
Hull        -0.86077397 -0.85571938
Liverpool   -0.31488666 -0.11875535
Man City    -0.12473511 -0.38284408
Man United   0.02234448  0.07978077
Portsmouth  -0.86981651 -0.72757785
Stoke       -0.88764279 -0.40852499
Sunderland  -0.53414397 -0.57689704
Tottenham   -0.21482565 -0.28337794
West Ham    -0.54516113 -0.74067321
Wigan       -0.77192547 -0.91086544
Wolves      -0.94049578 -0.56096282

$home
[1] 0.4579831
```

As expected from looking at the final 2009-2010 league table, only Chelsea and Manchester United have attack strengths larger than Arsenal. Both teams also have defence strengths larger than Arsenal's, which also makes sense as they finished higher in the table. However Aston Villa, Liverpool and Tottenham also have larger defence strengths despite being lower in the table. This will be because they have a better defence than Arsenal but not as good an attack and therefore Arsenal's superior attack strength must outweigh their inferior defence strength and hence they came third. The home advantage parameter shows it's significance by being comfortably above 0 meaning that it has a large effect on the scores.

Chapter 5

Simulating Games

5.1 Simulating One Game From the Parameters

The set of parameters estimated in the previous chapter open up many new avenues of statistics to analyse. To calculate the parameters, the scores from games, Y, were used. Now these parameters can be used to simulate scores of games. By setting up the matrix X for the games to be simulated and putting the set of 41 parameters into order and named as the vector $\boldsymbol{\beta}$, they can inserted into the formula $Y = \exp(X\boldsymbol{\beta})$ as in 3.3.3 and the vector Y is calculated. This vector will be formed as shown in equation 4.3.4:

$$Y = (\exp(\alpha_a - \gamma_b + \delta), \exp(\alpha_b - \gamma_a), ...).$$

As discussed in section 4.2 these values contained in the vector are λ values of a Poisson distribution for the scores.

Example

Say a game between Arsenal and Wolves is to be simulated with Arsenal playing at home. Parameters calculated from the 2009-2010 Premiership season can be used from the end of the previous chapter. So using these parameters and equation 5.1.1, Arsenal's score would have

$$\begin{aligned} \lambda_{Ars} &= \exp(\alpha_{Ars} - \gamma_{Wol} + \delta) \\ &= \exp(0 - (-0.56096282) + 0.4579831) \\ &= 2.77027313... \end{aligned} \tag{5.1.1}$$

and Wolves' score would have

$$\begin{aligned}\lambda_{Wol} &= \exp(\alpha_{Wol} - \gamma_{Ars}) \\ &= \exp(-0.94049578 - (-0.29966558)) \\ &= 0.52685484... \end{aligned} \tag{5.1.2}$$

meaning that Arsenal look very likely to win the game as their expected number of goals is much higher than Wolves'. This expected number of goals is because $E(X) = \lambda$ where X is the number of goals scored by a team and is proved in section 2.2.2. Obviously the score in the game cannot be 'Arsenal 2.77... – 0.52... Wolves' as this is impossible with count data. Therefore, as the two values are λ values for two different Poisson distribution, the R function `rpois` can be used to give one random value from each of these distributions as shown below.

```
> Arsenal <- rpois(1,lambdaArs)
> Wolves <- rpois(1,lambdaWol)
> Arsenal
[1] 2
> Wolves
[1] 0
```

This is where `lambdaArs` $= \lambda_{Ars}$ and `lambdaWol` $= \lambda_{Wol}$ are calculated in R in the same way as equations 5.1.1 and 5.1.2. This particular random generation of a score gives a result of Arsenal 2-0 Wolves. By looking at the distributions for each value of λ for each team in figure 5.1, it is clear that this is in fact the most likely result as $P(X = 2)$ is highest for Arsenal and $P(X = 0)$ is highest for Wolves. If a large number of values are randomly generated, i.e. `rpois(n,lambda)` for large n and specified value of λ, then frequency bar charts are drawn of the results, for the λ values for Arsenal and Wolves, the bar charts would look very similar to the probability mass functions in figure 5.1. As $n \to \infty$, the frequency chart will tend to the pmf (equation 2.2.1).

So to simulate a single game, the code above can be used as it involves working out the expected number of goals for each team given who they are playing against. This gives a general knowledge of what the score is expected to be. Then the `rpois` function gives a random value from each distribution of expected values to give a simulated score. This random selection represents the randomness in sport which is very common. More often than not, Arsenal will come out winners of this game due to their higher λ value.

The graphs (figure 5.1) show the expected probabilities of each team scoring a particular number of goals. Therefore, it is possible it calculate the probability of each result happening. For example, to work out the probability of the score Arsenal 0-0 Wolves, the formula $P(Ars = 0 and Wol = 0) = P(Ars = 0) \times P(Wol = 0)$ can be used. This is the probability of Arsenal scoring 0 goals and Wolves scoring 0 goals both happening. Independence is assumed here so that the formula

$$P(AB) = P(A)P(B)$$

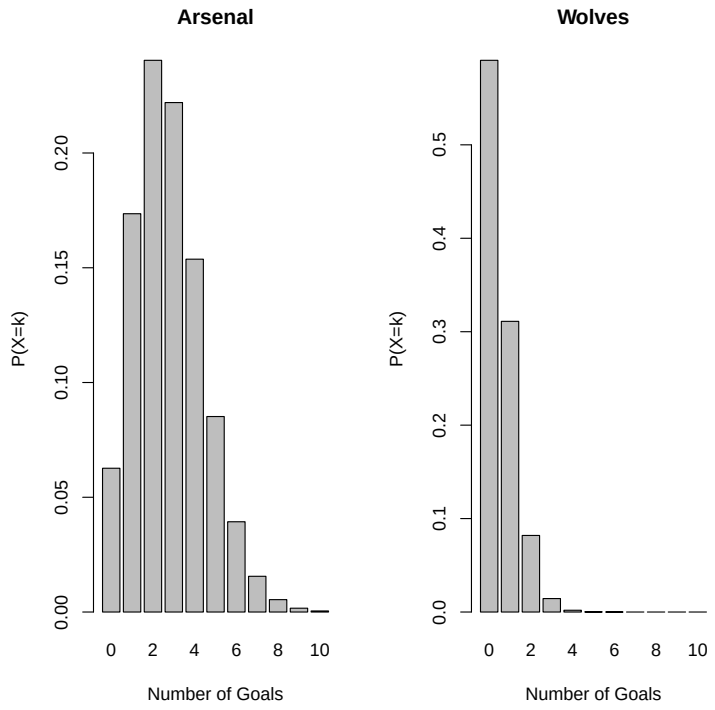

Figure 5.1: Probability mass functions for Arsenal and Wolves expected number of goals

can be used to multiply the two probabilities together. Obviously there is going to be some dependence in real life on how many goals the teams score as they are playing against each other. However for the purpose of simulating games from the model, independence is assumed.

Using this for every result, a probability table can be formed to show all the probabilities of different results in one table. A probability table for the game 'Arsenal vs Wolves' is shown below, built using the R function in appendix C.

```
> Probabilities <- ProbTable(TeamParameters,"Arsenal","Wolves")
> Probabilities
      0     1     2     3    4    5    6   7+
0  3.70 10.25 14.19 13.11 9.08 5.03 2.32 1.37
1  1.95  5.40  7.48  6.91 4.78 2.65 1.22 0.72
2  0.51  1.42  1.97  1.82 1.26 0.70 0.32 0.19
3  0.09  0.25  0.35  0.32 0.22 0.12 0.06 0.03
4  0.01  0.03  0.05  0.04 0.03 0.02 0.01 0.00
5  0.00  0.00  0.00  0.00 0.00 0.00 0.00 0.00
6  0.00  0.00  0.00  0.00 0.00 0.00 0.00 0.00
7+ 0.00  0.00  0.00  0.00 0.00 0.00 0.00 0.00
```

So the most likely score according to this probability table is Arsenal 2-0 Wolves with probability 14.19%. This was already known from the two graphs 5.1 showing the probability mass functions

of the two team's expected number of goals. Also, the second most likely score is 3-0 to Arsenal (13.11%) and the third 1-0 to Arsenal (10.25%). From general football knowledge these results do make sense as Arsenal are one the of strongest teams in the Premiership and Wolves are one of the weakest. Now the probabilities of each score can be used to calculate the probability of each result, i.e. home win, draw and away win, also using an R function shown in appendix D. This works by adding all of the probabilities of results where Arsenal win together and saying that is the probability of Arsenal winning. The same is done for a draw and a Wolves win.

```
> ResultProbabilities <- ResultProbs(Probabilities)
> ResultProbabilities
$HomeWin
[1] 83.86

$Draw
[1] 11.42

$AwayWin
[1] 4.7
```

Therefore Arsenal have a probability of winning of 83.86%, there is a 11.42% chance of there being a draw between the two teams and Wolves have a 4.7% chance of winning. Obviously these probabilities are not exact because they are only based on the strengths on the two teams from one season's data and anything can happen on the day on a football match. However they do give an indication of how the game is likely to end up.

5.2 Simulating a Whole Season From the Parameters

If one game can be simulated using the estimated parameters as shown in the previous section (5.1) then a whole season's worth of games can also be simulated. To do this, all possible games between the n teams in the parameter table need to be created and simulated one at a time. Again, a function built in R can produce a set of games for a full season using an input of a parameter table e.g. `TeamParameters`. The function, named `Games` as shown in appendix E, takes the set of parameters and creates the full set of games from the n teams in the parameter table. This is done by taking the list of teams in the parameter table and giving each of the n teams $n-1$ home games against each of the other $n-1$ teams and therefore including each team's away fixture against each of the other $n-1$ teams as well. The `if` statement in the function makes sure a team cannot play itself.

So these n sets of $n-1$ games (equating to $n \times (n-1)$ games overall) are created as rows of a data frame one at a time and simulated as in the previous section. Therefore a data frame will be outputted in the same format as `results0910` with $n \times (n-1)$ rows representing that number of games. So for the example of a Premiership season with 20 teams there would be 380

games outputted. These games would have the first alphabetical team's home games in the first $n-1$ rows and each team's $n-1$ home games in alphabetical order there after.

Example

Using the parameters `TeamParameters` from the end of section 4.4.1 and inputting these into the R function `Games`, a whole season can be simulated.

```
> SimSeason <- Games(TeamParameters)
> SimSeason
             V1          V2 V3 V4
1       Arsenal Aston Villa  1  0
2       Arsenal  Birmingham  4  2
3       Arsenal   Blackburn  4  0
...
378      Wolves   Tottenham  0  3
379      Wolves    West Ham  1  0
380      Wolves       Wigan  2  0
```

So these 380 games simulate a Premiership season with the same teams in it as in the 2009-2010 Premiership season from which the parameters were estimated. As pointed out, the first team alphabetically, Arsenal, have their games first and so on until the last team alphabetically, Wolves, have their home games at the end of the list. Each game is simulated as shown in section 5.1 and gives a full set of results for the season. This is the same set of games as the 2009-2010 Premiership season but with simulated results rather than the real life results, so by creating a table using the function `Table` (appendix A) as used before to create a league table from a set of results, it is possible to compare the simulated season with the real one.

```
> SimTable <- Table(SimSeason)
> SimTable
          Team  P HW HD HL HF HA AW AD AL AF AA  GD Points
1   Man United 38 18  1  0 55 12 10  2  7 32 20  55     87
2    Tottenham 38 14  3  2 41 14 13  3  3 33 14  46     87
3      Chelsea 38 16  2  1 59 15  8  8  3 39 24  59     82
4      Arsenal 38 14  4  1 45 20  8  7  4 37 25  37     77
5  Aston Villa 38 11  4  4 30 16 10  2  7 21 24  11     69
6    Liverpool 38 11  5  3 37 18  9  3  7 31 24  26     68
7      Everton 38 10  4  5 37 25  7  4  8 29 27  14     59
8       Fulham 38 12  6  1 23  8  2  4 13 10 31  -6     52
9     Man City 38 10  4  5 32 20  4  3 12 24 33   3     49
10   Blackburn 38 10  3  6 25 24  5  1 13 11 26 -14     49
11      Bolton 38 13  3  3 24 12  1  4 14 10 38 -16     49
12  Sunderland 38  9  4  6 24 19  3  6 10 20 36 -11     46
13  Birmingham 38  7  7  5 14 15  3  5 11 11 26 -16     42
14       Stoke 38  7  4  8 16 20  3  6 10 15 27 -16     40
```

```
15  Portsmouth 38  6  2 11 22 28  4  4 11 11 26 -21     36
16    West Ham 38  7  4  8 30 32  2  4 13 10 37 -29     35
17      Wolves 38  7  4  8 20 26  1  6 12 12 33 -27     34
18        Hull 38  6  8  5 18 21  1  5 13 12 39 -30     34
19       Wigan 38  5  6  8 21 27  3  3 13 19 48 -35     33
20     Burnley 38  5  5  9 20 28  2  3 14 13 35 -30     29
```

There are a few key differences that can be seen between this simulated season and the real season. One is that Tottenham have finished in second place, ahead of Chelsea and Arsenal. This is not necessarily expected because Tottenham are not viewed to be good enough to achieve this league position, however, perhaps due to good attack and defence strengths, over this simulated season, they have managed to finish second. Two more large differences are that Manchester City finished 9th compared to their 5th place finish in the real table and Portsmouth finished 15th[1] compared to their real finish of 20th.

Different finishing positions of each team between the real and simulated seasons will be mainly because of the random value taken to simulate a team's score in each match. Although the parameters represent the strengths of the teams compared to each other, there is still room for unexpected results. This randomness however, does try and enact what happens in real life where there are many twists and turns that happen that will not occur in a simulated season. For example, if a team went on a long cup run meaning they had a lot a games to play and faltered in the league consequently, this would not be taken into account in the simulated environment. When trying to simulate a whole season at one time, this cannot be taken into account, along with other variables that can occur during a season. These variables can include: cup runs (as mentioned); injuries to key players; difference in form (also mentioned) which can also result in pressure on a team; and many others. None of these variables that all happen within a season will, or can, be taken into account when simulating a season all in one go. This is because when a season is simulated as above, all games are effectively taking place at the same time.

The next step is to look at how a team is affected from season to season, rather than during the season. One of the main variables that affects teams between seasons is the transfers of players in and out which inevitably changes a team's strength. This could mean a team is likely to finish higher or lower than the previous season depending on the transfers. Is this the case, or is the difference in a team's final league position from season to season just a random fluctuation?

5.3 League Table Randomness

To test whether there are random fluctuations between seasons, instead of just simulating one season from the parameters estimated using the 2009-2010 Premiership season, simulate numerous seasons and then look at the statistics of where the 20 teams have finished in each season. To do this, another function built in R can be used called `Sim` (appendix F). This function takes

[1] With Portsmouth's 9 point reduction that they incurred during the season, in the simulated season they would finis with 27 points and finish 20th but this can be ignored for the purpose of analysing the simulation.

the parameters as an input, along with a specified number of simulations to process (k) and simulates k league tables one at a time as shown in the previous section (5.2).

A `for` loop is used to do each simulation and the league positions of the teams are stored in a data frame after each simulation. The output is a data frame with each row representing one of the teams and each column representing each simulation and giving the position a team finished in that simulation. To get a large enough set of results to give a good representation for the statistics, let $k = 1,000$ which should cancel out any extreme randomness that occurs.

```
> Simulations <- Sim(TeamParameters, 1000)
> Simulations
...
```

	V988	V989	V990	V991	V992	V993	V994	V995	V996	V997	V998	V999	V1000
Arsenal	3	4	2	5	2	3	4	2	4	7	3	7	5
Aston Villa	6	6	6	6	4	5	8	8	6	3	6	9	8
Birmingham	10	14	13	17	14	12	10	11	18	12	11	8	11
Blackburn	16	13	9	10	15	14	12	12	12	14	12	17	14
Bolton	17	12	12	12	17	15	7	16	13	19	13	11	17
Burnley	13	18	14	20	18	19	14	13	17	18	14	12	19
Chelsea	1	1	1	2	1	1	1	3	1	2	1	1	1
Everton	12	7	8	7	7	8	9	7	7	9	4	4	9
Fulham	15	11	11	9	8	10	17	10	10	10	15	14	10
Hull	19	20	18	13	20	20	20	20	19	20	17	18	20
Liverpool	5	8	10	4	5	6	5	5	3	4	5	2	3
Man City	4	3	3	3	6	4	3	4	8	5	8	3	7
Man United	2	2	4	1	3	2	2	1	2	1	2	5	2
Portsmouth	11	10	20	18	19	16	19	15	14	15	20	19	12
Stoke	8	17	7	15	11	9	15	14	16	13	16	13	16
Sunderland	14	9	16	14	10	11	11	9	15	6	9	10	13
Tottenham	7	5	5	8	9	7	6	6	5	8	7	6	4
West Ham	9	16	17	11	13	13	13	17	9	11	10	16	6
Wigan	18	19	15	16	12	17	16	19	20	17	19	20	15
Wolves	20	15	19	19	16	18	18	18	11	16	18	15	18

Simulation Statistics

So now the 1,000 simulations have been implemented, some statistics can be calculated about each team's set of finishing positions. The statistics used for each team are average position along with mode position to give an idea of where a team is likely to finish and also and standard deviation of the 1,000 positions for each team which gives the spread of where the results. Standard deviation is the statistic that will indicate the randomness of the league table. To calculate the statistics (`SimulationStats`) another R function can be used called `SimStats` (appendix G) where simulations are inputted and statistics about each row (team) are outputted.

```
> SimulationStats <- SimStats(Simulations)
```

```
> SimulationStats
          Team Average     StDev Mode      Attack     Defence
1      Chelsea   1.500 0.7102848    1  0.20736792 -0.07159737
2   Man United   1.965 0.9448109    2  0.02234448  0.07978077
3      Arsenal   3.673 1.4512674    3  0.00000000 -0.29966558
4    Liverpool   5.156 1.7494927    4 -0.31488666 -0.11875535
5     Man City   5.159 1.7101590    5 -0.12473511 -0.38284408
6    Tottenham   5.365 1.8174372    5 -0.21482565 -0.28337794
7  Aston Villa   7.081 1.9349892    7 -0.47087219 -0.21825368
8      Everton   7.458 2.0075523    8 -0.31741611 -0.45507449
9   Sunderland  11.131 2.6920412    9 -0.53414397 -0.57689704
10      Fulham  11.214 2.7069414    9 -0.75218030 -0.37080784
11  Birmingham  11.836 2.8564424    9 -0.77721659 -0.39136711
12   Blackburn  12.604 2.9287816   11 -0.69316106 -0.55183181
13    West Ham  13.052 2.9250388   13 -0.54516113 -0.74067321
14       Stoke  13.110 2.8927959   13 -0.88764279 -0.40852499
15      Bolton  14.704 2.8132363   14 -0.65696644 -0.75069397
16      Wolves  15.421 2.8432100   18 -0.94049578 -0.56096282
17  Portsmouth  16.708 2.5524990   18 -0.86981651 -0.72757785
18     Burnley  17.083 2.4446037   19 -0.64168730 -0.95337217
19       Wigan  17.814 2.1489584   20 -0.77192547 -0.91086544
20        Hull  17.966 2.2139887   20 -0.86077397 -0.85571938
```

These statistics show some interesting facts, some of which confirm what general knowledge about football would conclude. Firstly, looking at the average position for each team, this column indicates that there is a clear top 8 in the Premiership because Everton in 8^{th} have an average position of ≈ 7.5 whereas the team one behind them, Sunderland in 9^{th}, have an average position of ≈ 11.1 which is almost 4 positions lower. This has generally been the case in real life football over the last few years so this does seem to show that the parameters estimated are representative of the teams' strengths. However, the mode values for the teams (i.e. most common position) makes for different reading because the top 11 teams have modal values of 9 or lower. There are also big differences between average and mode values towards the bottom of the table. Some of these will be analysed in section 5.3.1.

The lower standard deviations of these top 8 teams confirms that over the 1,000 simulations they are fairly certain to finish in these top 8 places. Standard deviation values of less than 1 for Chelsea and Manchester United show that they are very likely to finish in the top two due to superior strengths. Again this had been the case over recent years that these two teams have been finishing in the top two positions. Both teams are quite similar in strength however, which could indicate some randomness between who finishes in 1^{st} position with the odd random year being that neither win the league.

What is interesting though is that the standard deviations of the teams are larger in the bottom half. So when looking at the randomness of the table, it is clear that it is fairly certain who is going to come in the top 2-3 positions, then quite certain who will finish in positions 4-8. However, from positions 9-17, each of the teams had a standard deviation of between 2.5-3. This

means, assuming the positions of a team are normally distributed around the mean, that the teams in the botom half of the table are more likely randomly positioned.

The normal distribution means that one of these teams with a standard deviation of 2.5, say, has a probability of $\approx 68\%$ of being one standard deviation (i.e. 2-3 positions either side of the mean) away form the mean. Also, a team with standard deviation of 2.5 would have a probability of $\approx 95\%$ of coming 5 positions either side of their mean. Ranges of this size show that these teams cannot be certain of where they are going to finish. Figure 5.2 shows how the bottom half of the table can have a much more random finish to it than the top half due to the high standard deviations in the lower teams. This will probably be because the teams are well matched and have similar strengths to each other. Whereas the top teams are significantly higher giving a structure to the league with a certain aspect of randomness too.

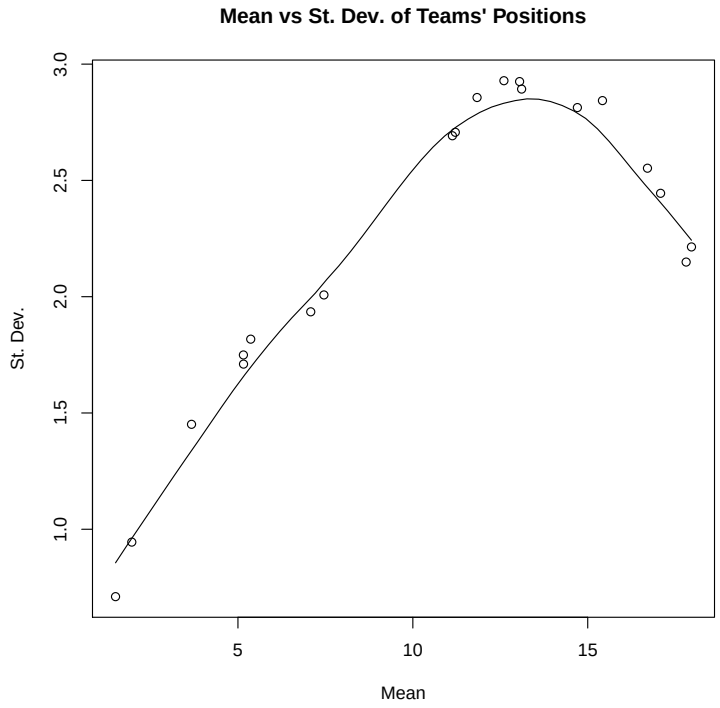

Figure 5.2: Mean position against Standard Deviation of positions from set of simulations

The main areas of interest in the Premiership are the top 6 places (the winner and European qualification places) and the bottom 3 places (relegation to the lower division. Although the top 6 can be predicted, the order in which they finish cannot due to the natural randomness in football. The bottom three places are very difficult to predict. As figure 5.2 shows, there are 4 teams with a slightly lower standard deviation than the 8 teams in the rankings above them suggesting those 4 teams more likely to be in the bottom 4. However still with standard deviations between 2 and 2.5, there is room for variation. This is because the teams seem to be very similar in strength in the bottom half and so it would only take a good season from one

team and a bad season by another to change the table around. As the parameters show though, the bottom three teams in the simulations do have the worst defensive parameters by a decent margin and looks like this is why they are in the relegation zone.

```
> SimulationStats
                Team Average    StDev Mode       Attack     Defence
1          Newcastle    1.48 1.114460    1  0.505509767  0.53805517
2          West Brom    2.67 1.657886    2  0.503946292  0.22230569
3             Forest    4.63 2.798466    3  0.183244581  0.42260697
4            Cardiff    6.29 3.830486    4  0.309772467  0.11612533
5          Blackpool    6.90 3.421560    5  0.326370606  0.04376507
6          Leicester    6.90 3.328785    8  0.123273454  0.30756345
7      Middlesbrough    8.92 4.113737    7  0.076403733  0.20422765
8  Sheffield United    9.48 4.423263    7  0.146859316  0.10583578
9            Reading   10.38 4.644493    5  0.245322536 -0.03467123
10         Doncaster   11.24 4.463364    8  0.099370717  0.05482786
11           Swansea   11.35 4.689070   12 -0.304914488  0.51860797
12   Crystal Palace   13.30 5.201981   22 -0.070054022  0.15168496
13              QPR   13.73 4.800789   15  0.087387122 -0.05860225
14          Watford   14.81 5.147707   16  0.140141761 -0.10603504
15     Bristol City   14.89 5.065022   14  0.052225517 -0.05713019
16            Derby   15.47 4.368054   13 -0.004405411 -0.02361717
17          Ipswich   15.54 5.277013   11 -0.064235617  0.01089174
18          Preston   16.55 3.916747   19  0.093299103 -0.17492727
19         Barnsley   17.27 4.437797   21  0.000000000 -0.11476006
20         Coventry   18.08 4.665974   23 -0.124021960 -0.03500610
21       Scunthorpe   18.42 4.190369   21  0.168370331 -0.31863481
22   Sheffield Weds   19.56 3.712714   22 -0.078620351 -0.11182089
23         Plymouth   20.48 3.696791   24 -0.210191650 -0.09280829
24        Peterboro   21.66 3.029218   24 -0.133825247 -0.25781574
```

Interestingly, looking at the stats for the Championship where the parameters are estimated from the 2009-2010 season and 100 simulations are calculated, the standard deviations are a lot higher. This could be because the teams are much more evenly matched than in the Premiership and so could finish in a larger range of positions. Crystal Palace have a very intriguing statistic that their average position puts them in 12^{th} position whereas their most common position is 22^{nd} which probably means they are equally as likely to come in most positions. It does seem to be fairly certain however that Newcastle and West Brom are going to finish in the top two and Sheffield Wednesday, Plymouth and Peterborough are likely to be the bottom three teams due to their lower standard deviation values.

5.3.1 Team Statistics

To look at some of the statistics a bit more closely, take the two teams who have been analysed before of Arsenal and Wolves and also look at Sunderland who will give a mid-table team to

compare with as well. Each of their sets of 1,000 positions from the simulation are shown in figure 5.3 so it can be seen where their statistics came from. Arsenal, Sunderland and Wolves are two very different teams so will give a good comparison between the types of team. It is actually Wolves' statistics that are very interesting. Although the team's mean finishing position is 15.421 which means, on average, they would avoid being relegated from the league and this average puts Wolves in 15^{th} in the ranking but Wolves' modal value is actually 18 meaning that their most common finishing is in the relegation places. Fortunately for Wolves, this modal value is only slightly more frequent than the number of times the team finished 15^{th}, 16^{th} or 17^{th}.

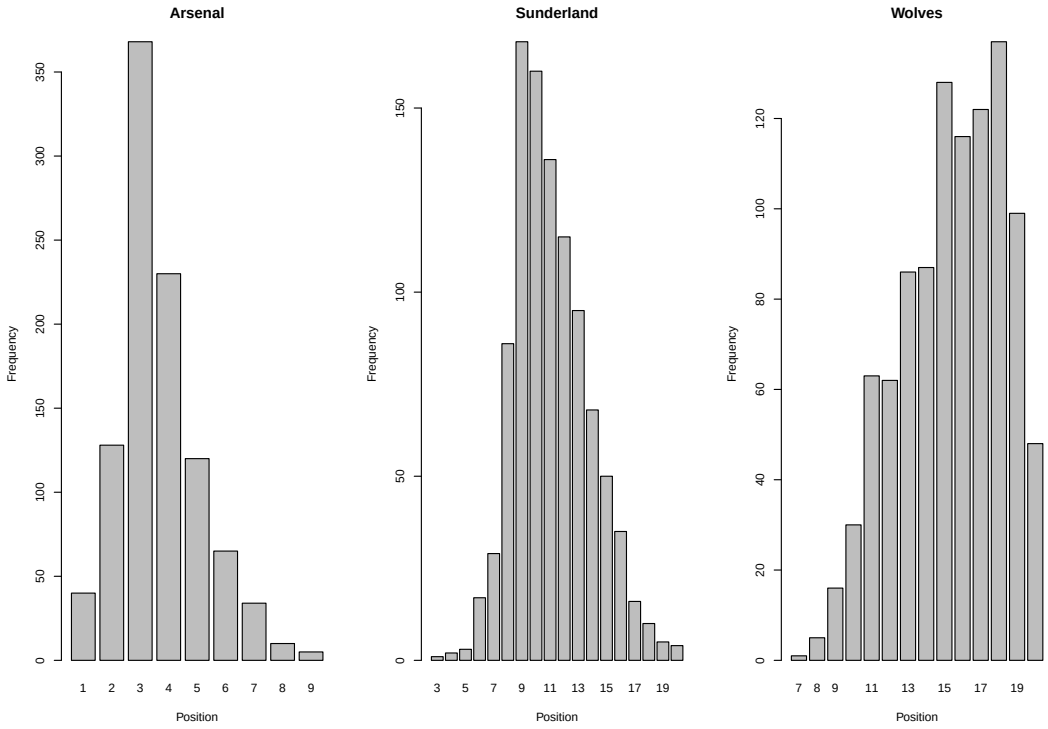

Figure 5.3: Position frequencies from the 1,000 simulations

These graphs show the difference in range of positions between the top and bottom halves of the league. Arsenal, whose mean value is 3.673 and modal value is 3, have a very small range (1^{st} to 9^{th}) of positions showing that it is fairly certain in what area of the table they will finish.

Sunderland however, who are ranked 9^{th} in the simulation statistics, have a huge range of between 3^{rd} and 20^{th}. This shows that Sunderland, whilst being a reasonably strong team, are susceptible to finishing lower in the table. Their modal finish is 9^{th} whereas their mean position is actually 11.131. This is represented in the the figure though (figure 5.3) because Sunderland also finished 10^{th}, 11^{th} and 12^{th}. Random fluctuations from season to season could result in the spread of finishing positions, along with all of the other teams around Sunderland. As these teams are all so closely matched as the stats suggest, randomness is a logical explanation.

Wolves have a similar range but only finished a highest position of 7^{th}. However, the frequencies

of their finishes a lot more spread out. By looking at the y-axis of each team's graph, it can be seen that Wolves finish in different positions a larger number of times. Their modal position is 18^{th} but positions 15, 16 and 17 were all achieved a similar number of times. Clearly the three team's standard deviations have the main effect on the spread here but it does depend on how strong the team is as to whether they have a large or small range of possible positions.

5.4 Parameter Accuracy

Simulations have been carried out using the estimated parameters but are these parameters accurate? To test whether they are, take the original parameters and use the `Games` function (appendix E) to simulate a set of games from these parameters. If this is done k times, say, from the original parameters and new parameters are estimated from each set of simulated games using the function `Parameters` (appendix B) then an statistics of these k sets of parameters can be taken. This gives a set of bias and standard deviation values for each team's k parameter estimations. The bias is calculated using the formula

$$\text{bias} = \mathbb{E}(\theta - \hat{\theta})$$

where θ is the original parameter estimated and $\hat{\theta}$ is the average of the set of k parameters. So the bias calculates the difference between the original parameters and the average of the estimated parameters. The standard deviations are calculated using the usual formula

$$s = \frac{n}{n-1} sqrt\bar{x^2} - \bar{x}^2$$

for standard deviation. An R function `MultiPara` (appendix H) can be used and the original parameters inputted to output the statistics about the parameters. To eradicate any errors, choose the number of sets of parameters to be estimated to be large (1,000 say). Then in R, this can be implemented as below.

```
> MultiParameters <- MultiPara(TeamParameters,1000)
> MultiParameters
$teams
             Attack.bias Attack.sd  Defence.bias Defence.sd
Arsenal      0.000000000 0.0000000 -0.0089995629  0.2030080
Aston Villa  0.005597228 0.1793082 -0.0182264683  0.1980455
Birmingham   0.019092982 0.1922822 -0.0155696513  0.1865079
Blackburn    0.017634342 0.1951727 -0.0103789954  0.1828515
Bolton       0.009288657 0.1946217 -0.0088078697  0.1701315
Burnley      0.010600368 0.1969018 -0.0005486783  0.1596348
Chelsea      0.006629344 0.1492916 -0.0206157729  0.2170217
Everton      0.002806169 0.1739145 -0.0150988317  0.1866231
Fulham       0.005503256 0.2012078 -0.0220704375  0.1919449
Hull         0.011737158 0.2121938 -0.0085660405  0.1651465
Liverpool    0.003280723 0.1733819 -0.0141952641  0.2109052
```

```
Man City    -0.001420856 0.1625654 -0.0122604157  0.1923252
Man United   0.001110368 0.1567345 -0.0277248375  0.2284775
Portsmouth   0.004264397 0.2105170 -0.0119166144  0.1703035
Stoke        0.006672052 0.2008677 -0.0187160499  0.1875453
Sunderland   0.008956824 0.1806824 -0.0068197871  0.1749779
Tottenham    0.003115948 0.1710935 -0.0116028601  0.1946505
West Ham     0.012261710 0.1822414 -0.0102045820  0.1758287
Wigan        0.005134414 0.2001824 -0.0130726713  0.1628383
Wolves       0.002150422 0.2082838 -0.0068289441  0.1728925

$home.bias
[1] 0.0002660008

$home.sd
[1] 0.0638572
```

The standard deviations for each parameter shown here are very small, in fact all are below 0.25. This shows that the set of parameters estimated using the original ones are all very close to the original estimated parameters. The bias values also show this because they are all very close to 0, showing that very little bias has occurred in the 1,000 simulated set of parameters. This gives a lot of confidence that the parameter estimation method used has been successful and does represent the real strengths of the teams and home advantage very well. Another way to show this is the rankings table (5.1).

The rankings table (5.1) shows that the parameters and the simulations do follow what happened in real life so the Poisson regression model seems to have been the correct method to use. It is quite interesting to see however, that some teams clearly overachieved and some underachieved in the 2009-2010 season. By comparing the 09-10 final position to that of their simulated position and strengths, Sunderland and West Ham are the two teams that have underachieved the most. This could have been because of their much lower defensive ranking. Although their lower finish may have just been random fluctuations in the league. Fulham also seemed to have underachieved and they will hope that their superior defensive strength will mean they finish higher in the future. The main overachievers are Birmingham whose defensive strength seems to have helped a lot in this season. Other than these stand out teams though, apart form random fluctuations in the league that the statistics show seem to occur, the simulations and parameter estimations do closely follow the real life table.

Team	09-10	Simulated	Attack	Defence	Average Parameter
Chelsea	1	1	1	2	1.5
Man United	2	2	2	1	1.5
Arsenal	3	3	3	6	4.5
Tottenham	4	6	5	5	5
Man City	5	5	4	8	6
Aston Villa	6	7	8	4	6
Liverpool	7	4	6	3	4.5
Everton	8	8	7	11	9
Birmingham	9	11	16	9	12.5
Blackburn	10	12	13	12	12.5
Stoke	11	14	19	10	14.5
Fulham	12	10	14	7	10.5
Sunderland	13	9	9	14	11.5
Bolton	14	15	12	17	14.5
Wolves	15	16	20	13	16.5
Wigan	16	19	15	19	17
West Ham	17	13	10	16	13
Burnley	18	18	11	20	15.5
Hull	19	20	17	18	17.5
Portsmouth	20	17	18	15	16.5

Table 5.1: Rankings of the 20 2009-2010 Premiership teams for: the 09-10 real season; the average position in the simulations; attack and defence strengths; average ranking of attack and defence.

Chapter 6

Predicting a Future Table

Now it is known that the parameter estimation is done to a good degree of accuracy, it is the next step to try and predict results and tables. The 2010-2011 Premiership season has not finished yet so trying to predict the whole season from the beginning will mean the resulting table can be compared to what is actually happening. However, just taking the results from the 2009-2010 Premiership season to predict the 2010-2011 season is not good enough as there are 3 new teams who were promoted from the Championship. Therefore, just using the data form the 2009-2010 premiership will not give parameters for these 3 new teams. To solve this, the 2009-2010 Championship data is needed. This is still not enough because the 3 promoted teams do not have any comparison to the Premiership teams and will therefore be very high which will clearly not be the case. To combat this, there needs to be some crossover between the leagues so if both the 2008-2009 and 2009-2010 seasons for both leagues are taken into account then there will be teams who have played against everyone (i.e. whoever got relegated form the Premiership in 2008-2009 and whoever got promoted form the Championship in the same year).

There are 24 teams in the Championship so it will contain $24 \times 23 = 552$ games in a season. Therefore the data set named `results080910` will contain $(2 \times 380) + (2 \times 552) = 1864$ games. Taking all these 4 seasons worth of data means that there will be parameters for 47 teams, 20 from the Premiership, 24 from the Championship and 3 extra for the teams who were promoted to the Championship in 2008-2009. The set of results is inputted into the `Parameters` function in R below and the output is shown.

```
> TeamParameters1 <- Parameters(results080910)
> TeamParameters1
$teams
                     Attack     Defence
Arsenal           0.00000000 -0.27716147
Aston Villa      -0.35010151 -0.36276063
Barnsley         -1.05174343 -1.03212219
Birmingham       -0.86960688 -0.46093337
Blackburn        -0.60504299 -0.62934185
```

```
Blackpool        -0.84468144 -0.95005142
...
Watford          -0.77135057 -1.14159310
West Brom        -0.53677108 -0.77396438
West Ham         -0.51271559 -0.59806728
Wigan            -0.73240131 -0.69961198
Wolves           -0.66259038 -0.72098259

$home
[1] 0.3291451
```

As can be seen, parameters for the Championship teams (Barnsley, Blackpool, Watford and West Brom and more that are not shown to save space) have smaller parameters than the others. This is because Newcastle, Middlesbrough and West Brom have played in the Premiership in 2008-2009 and the Championship in 2009-2010 so have played everyone giving a comparison between the two leagues.

6.1 Predicting the 2010-2011 Premiership Season

The set of parameters estimated now contain parameters for the 20 teams competing in the 2010-2011 Premiership season. These are the same as the 2009-2010 season but without Burnley, Hull and Portsmouth and with Newcastle, West Brom and Blackpool. So to simulate the current 2010-2011 season, input the parameters above (`TeamParamters1`) into the function `Games` (appendix E) and create a table using the function `Table` (appendix A). The change that needs to be made is that the second line in the `Games` function changes to

```
teams <- c("Arsenal", "Aston Villa", "Birmingham", "Blackburn", "Blackpool",
   "Bolton", "Chelsea", "Everton", "Fulham", "Liverpool", "Man City",
   "Man United", "Newcastle", "Stoke", "Sunderland", "Tottenham",
   "West Brom", "West Ham", "Wigan", "Wolves")
```

so that only the parameters for these teams involved in the 2010-2011 season are used. Then the predicted table is given below.

```
> PredTable <- Table(Games(TeamParameters1))
> PredTable
          Team  P HW HD HL HF HA AW AD AL AF AA  GD Points
1   Man United 38 14  1  4 32 10 12  5  2 35 11  46     84
2      Chelsea 38 15  3  1 56  9 11  2  6 40 25  62     83
3      Arsenal 38 13  1  5 29 17  9  7  3 33 19  26     74
4    Liverpool 38 11  6  2 30 10  9  5  5 24 16  28     71
5     Man City 38 10  6  3 30 12 11  2  6 35 27  26     71
6    Tottenham 38 11  2  6 31 21  7  5  7 19 25   4     61
```

```
7       Bolton 38  9  4  6 27 20  7  4  8 25 29   3     56
8  Aston Villa 38  9  4  6 26 15  7  3  9 21 23   9     55
9    Newcastle 38  9  5  5 23 14  4  6  9 17 28  -2     50
10      Fulham 38  8  5  6 25 21  5  6  8 15 22  -3     50
11  Sunderland 38  9  3  7 26 23  6  2 11 17 29  -9     50
12   Blackburn 38  9  2  8 23 22  5  4 10 17 33 -15     48
13      Wolves 38  5  5  9 22 27  8  2  9 15 23 -13     46
14     Everton 38  7  4  8 25 26  5  4 10 19 25  -7     44
15   West Brom 38  5  7  7 25 34  4  6  9 16 22 -15     40
16  Birmingham 38  7  3  9 18 25  3  6 10 12 27 -22     39
17       Stoke 38  7  4  8 19 26  3  5 11 12 30 -25     39
18    West Ham 38  6  6  7 24 27  3  5 11 17 34 -20     38
19       Wigan 38  5  5  9 17 29  4  3 12 11 28 -29     35
20   Blackpool 38  2  7 10 11 24  3  1 15 12 43 -44     23
```

To see if this prediction is close to what is actually happening in the league, the current 2010-2011 as of May 6^{th} 2011 is given below.

```
> Table1011 <- Table(results1011)
> Table1011
          Team  P HW HD HL HF HA AW AD AL AF AA  GD Points
1   Man United 35 16  1  0 43  9  5  9  4 28 24  38     73
2      Chelsea 35 14  2  2 37 11  7  5  5 29 17  38     70
3      Arsenal 35 11  4  3 32 13  8  6  3 36 23  32     67
4     Man City 34 11  4  2 30 12  7  4  6 23 19  22     62
5    Liverpool 35 12  4  2 37 12  4  3 10 17 27  15     55
6    Tottenham 34  8  8  1 27 17  6  5  6 23 26   7     55
7      Everton 35  7  7  3 28 22  4  8  6 20 21   5     48
8       Bolton 35 10  5  2 33 20  2  5 11 15 28   0     46
9       Fulham 35  8  6  3 26 16  2  9  7 17 20   7     45
10       Stoke 35  9  4  4 28 16  3  3 12 15 27   0     43
11   West Brom 35  7  6  5 29 30  4  4  9 22 35 -14     43
12   Newcastle 35  5  7  5 36 23  5  4  9 13 28  -2     41
13 Aston Villa 35  7  6  4 24 18  3  5 10 20 39 -13     41
14  Sunderland 35  7  5  6 24 24  3  6  8 15 28 -13     41
15  Birmingham 35  6  8  4 19 20  2  7  8 16 32 -17     39
16   Blackburn 35  7  6  5 21 15  3  2 12 20 40 -14     38
17   Blackpool 35  4  5  9 26 34  5  3  9 22 36 -22     35
18       Wigan 35  4  8  6 19 32  3  6  8 16 26 -23     35
19      Wolves 35  7  4  6 25 26  2  3 13 13 35 -23     34
20    West Ham 35  5  4  8 23 27  2  7  9 17 36 -23     32
```

By comparing both of the leagues, the prediction has actually given a close representation of what is taking place in real life. There are some differences like Aston Villa are not doing as well in real life and Stoke are doing a lot better but this will account for the randomness of the league table that can occur. The prediction has given a clear indication of which teams are

going to finish in certain positions. One of the main results is that it has predicted the top 6 correctly with only Liverpool and Manchester City interchanged and it has predicted two of the correct bottom 3 teams of West Ham and Wigan. There are still 31 games left to play in the Premiership however, so the real table is subject to change. Although it is unlikely to change dramatically, the predicted table is a good representation of the real life game.

6.2 Predicting the Remaining 2010-2011 Premiership Season

With 31 games remaining in the Premiership, it is possible to predict the results of these games. A slightly different method of simulating the results can be used to give the most likely results, rather than a random simulation from the λ values. This is done by using the probability table discussed in section 5.1 and picking out the highest probability as the most likely result in the match. So a slightly different version of the `Games` function can be used called `Games1` (appendix I) which gives the most likely result for a set of fixtures given. The R code below shows the results form the 2010-2011 Premiership season being inputted into R, along with the fixtures which are in the same format but without scores. Then parameters are estimated from the seasons results and the set of predictions of the most likely results are given as the output.

```
> fixtures <- read.table("Fixtures.csv", sep=",",stringsAsFactors=F)
> results <- read.table("Results1011.csv", sep=",", stringsAsFactors=F)
> TeamParameters <- Parameters(results)
> Predictions <- Games1(TeamParameters,fixtures)
> Predictions
            V1          V2 V3 V4         V5
1  Aston Villa       Wigan  1  0 0.11280577
2       Bolton  Sunderland  1  0 0.12266889
3      Everton    Man City  1  1 0.13428078
4    Newcastle  Birmingham  1  0 0.12656826
5     West Ham   Blackburn  1  1 0.11712360
6    Tottenham   Blackpool  2  1 0.09268980
7       Wolves   West Brom  1  1 0.09951460
8        Stoke     Arsenal  1  1 0.12705474
9   Man United     Chelsea  1  1 0.12557075
10      Fulham   Liverpool  1  0 0.14535411
11    Man City   Tottenham  1  0 0.14206406
12   Blackburn  Man United  0  1 0.10833538
13   Blackpool      Bolton  1  1 0.10015245
14  Sunderland      Wolves  1  0 0.11871878
15   West Brom     Everton  1  1 0.10939886
16     Chelsea   Newcastle  2  0 0.13147555
17     Arsenal Aston Villa  2  0 0.11250295
18  Birmingham      Fulham  0  1 0.15539653
19   Liverpool   Tottenham  1  0 0.11877566
20       Wigan    West Ham  1  1 0.12126930
```

```
21    Man City       Stoke  1  0 0.16757189
22 Aston Villa   Liverpool  1  1 0.12191029
23      Bolton    Man City  1  1 0.13182555
24     Everton     Chelsea  0  1 0.13436078
25      Fulham     Arsenal  1  1 0.13352383
26  Man United   Blackpool  3  0 0.09638827
27   Newcastle   West Brom  2  1 0.09453687
28       Stoke       Wigan  1  0 0.14893192
29   Tottenham  Birmingham  1  0 0.13738565
30    West Ham  Sunderland  1  1 0.12236317
31      Wolves   Blackburn  1  1 0.12073829
```

These are the most likely results of the remaining 31 Premiership games as of May 6^{th} 2011. The probabilities of each result are given as well. As this shows, the probabilities of these results are only around $10 - 15\%$ mainly. This is because any certain score is not very likely. However, it does give an indication of who is most likely to win each game and what the most likely outcome will be. Adding these results to the set of results that have already occurred, the table below shows a prediction of what the league will look like by the time the season has finished. These results can indicate whether a game is likely to finish as a home win, a draw or an away win by looking at what result appears in the table above.

```
> fullresults <- rbind(results,Predictions)
> PredTable1011 <- Table(fullresults)
> PredTable1011
          Team  P HW HD HL HF HA AW AD AL AF AA  GD Points
1   Man United 38 17  2  0 47 10  6  9  4 29 24  42     80
2      Chelsea 38 15  2  2 39 11  8  6  5 31 18  41     77
3      Arsenal 38 12  4  3 34 13  8  8  3 38 25  34     72
4     Man City 38 13  4  2 32 12  7  6  6 25 21  24     70
5    Tottenham 38 10  8  1 30 18  6  5  8 23 28   7     61
6    Liverpool 38 13  4  2 38 12  4  4 11 18 29  15     59
7       Fulham 38  9  7  3 28 17  3  9  7 18 20   9     52
8       Bolton 38 11  6  2 35 21  2  6 11 16 29   1     51
9      Everton 38  7  8  4 29 24  4  9  6 21 22   4     50
10       Stoke 38 10  5  4 30 17  3  3 13 15 28   0     47
11   Newcastle 38  7  7  5 39 24  5  4 10 13 30  -2     47
12  Sunderland 38  8  5  6 25 24  3  7  9 16 30 -13     45
13 Aston Villa 38  8  7  4 26 19  3  5 11 20 41 -14     45
14   West Brom 38  7  7  5 30 31  4  5 10 24 38 -15     45
15   Blackburn 38  7  6  6 21 16  3  4 12 22 42 -15     40
16  Birmingham 38  6  8  5 19 21  2  7 10 16 34 -20     39
17      Wolves 38  7  6  6 27 28  2  3 14 13 36 -24     36
18       Wigan 38  4  9  6 20 33  3  6 10 16 28 -25     36
19   Blackpool 38  4  6  9 27 35  5  3 11 23 41 -26     36
20    West Ham 38  5  6  8 25 29  2  8  9 18 37 -23     35
```

So using the predictions, Wigan, Blackpool and West Ham are going to be relegated to the Championship and Manchester United are going to win the league.

Chapter 7

Conclusion

Throughout this report, the Poisson regression model has been analysed to see whether it is a good method of estimating parameters for football teams. It is used to give each of the teams an attack and defence strength along with setting a home advantage strength too. By using football knowledge of the teams in real life, the parameters estimated (4.4.1) do represent the corresponding strengths of the teams and by using these parameters, games can be simulated to test whether they give sensible results. The statistics on the simulations (5.3) show that the simulated environment created by the parameter estimation does compare very closely to the real football data.

This means that the simulations can be used to assess the randomness of the league table in football. In football, teams tend to not finish in the same position each season and don't necessarily finish where they are expected to. The statistics about the simulations show that there is less randomness at the top of the Premiership than in the bottom half of the league. This may be because the teams who are better are much superior and will always come in the top positions, however towards the bottom of the league, teams are much more evenly matched and can therefore finish above and below each other very easily (see the statistics in section 5.3 to show this).

So the model fits the football data well, however, the parameters cannot be totally accurate. This is because a lot can happen in football that cannot be simulated. For example, when predicting a whole season, a cup run by a team or a serious injury to a key player can affect teams and these cannot be included in the parameter estimation. If each set of games were predicted one at a time during the season then the parameters could be adjusted accordingly but this is very difficult to get right as there is no way of telling how much something will affect a team.

Different ways of setting up the model could mean that the parameters are more accurate. For example, an away parameter could be added in to show the away strength, or weakness, of the teams. This would mean there were 42 parameters (41 for the `glm` function) and then a home team and away team's λ values would be calculated as follows instead.

$$\text{Home} \sim \text{Poiss}(\exp((\alpha_A + Home) - \gamma_B))$$

$$\text{Away} \sim \text{Poiss}(\exp(\alpha_B) - (\gamma_A + Away))$$

Also, even more parameters could be added. Attack and defence strengths for home and away for each team could be used to specify each team's home and away strength along with their attack and defence. There would then be no need for a home or an away parameter. However, there would then be 80 parameters because each team would have 4 each (attack, defence, home and away). This would just mean that X and $\boldsymbol{\beta}$ would be set up differently to incorporate the extra parameters. With only 380 games in a season though, 80 parameters is probably too many to estimate as there is not enough data to get a true representation. Another season could be added so the 760 games were used which could work and give more accurate parameters. Going any further back is risky though as teams do change a lot over 3 years.

The next steps to this project would be to try different sets of parameters in the model and seeing which give the best representation of real life. I would expect however, that the current method will be the most accurate because it is the simplest. Also, other leagues could also be analysed. The lower leagues in England could be analysed to a greater depth and also some European leagues could be looked into. The model should be able to analyse other sports too so this would be a very interesting idea to look into.

The results predicted in the report should not be used for betting purposes. They are only a representation of real life football and cannot be relied upon to give accurate predictions to future football matches. They do however give an excellent indication of what is likely to happen.

Appendix A

Creating a League Table

For this function, an input of a set of games is required in the same style as `results0910` in section 4.1. The output is a league table for the results entered of the teams involved.

```
Table <- function(games) {
    teams <- sort(unique(c(games[,1], games[,2])), decreasing = FALSE)
    n <- length(teams)
    g <- nrow(games)
    zero <- mat.or.vec(n, 1)
    T <- data.frame(Team=teams, P=zero, HW=zero, HD=zero, HL=zero, HF=zero, HA=zero,
                AW=zero, AD=zero, AL=zero, AF=zero, AA=zero, GD=zero, Points=zero)
    for (i in 1:g) {
        if (games[i,3] > games[i,4]) {
            T[which(teams == games[i,1]),"Points"] <-
                            T[which(teams == games[i,1]),"Points"] + 3
            T[which(teams == games[i,1]),"HW"] <-
                            T[which(teams == games[i,1]),"HW"] + 1
            T[which(teams == games[i,2]),"AL"] <-
                            T[which(teams == games[i,2]),"AL"] + 1
        } else {
            if (games[i,3] == games[i,4]) {
                T[which(teams == games[i,1]),"Points"] <-
                            T[which(teams == games[i,1]),"Points"] + 1
                T[which(teams == games[i,2]),"Points"] <-
                            T[which(teams == games[i,2]),"Points"] + 1
                T[which(teams == games[i,1]),"HD"] <-
                            T[which(teams == games[i,1]),"HD"] + 1
                T[which(teams == games[i,2]),"AD"] <-
                            T[which(teams == games[i,2]),"AD"] + 1
            } else {
                T[which(teams == games[i,2]),"Points"] <-
                            T[which(teams == games[i,2]),"Points"] + 3
                T[which(teams == games[i,2]),"AW"] <-
```

```
                        T[which(teams == games[i,2]),"AW"] + 1
                T[which(teams == games[i,1]),"HL"] <-
                        T[which(teams == games[i,1]),"HL"] + 1
            }
        }
        T[which(teams == games[i,1]),"P"] <- T[which(teams == games[i,1]),"P"] + 1
        T[which(teams == games[i,2]),"P"] <- T[which(teams == games[i,2]),"P"] + 1
        T[which(teams == games[i,1]),"HF"] <- T[which(teams == games[i,1]),"HF"] +
                                games[i,3]
        T[which(teams == games[i,1]),"HA"] <- T[which(teams == games[i,1]),"HA"] +
                                games[i,4]
        T[which(teams == games[i,2]),"AF"] <- T[which(teams == games[i,2]),"AF"] +
                                games[i,4]
        T[which(teams == games[i,2]),"AA"] <- T[which(teams == games[i,2]),"AA"] +
                                games[i,3]
        T[which(teams == games[i,1]),"GD"] <- T[which(teams == games[i,1]),"GD"] +
                                (games[i,3] - games[i,4])
        T[which(teams == games[i,2]),"GD"] <- T[which(teams == games[i,2]),"GD"] +
                                (games[i,4] - games[i,3])
    }
    S <-  data.frame(row.names=c(1:n), T[with(T, order(-Points, -GD)), ])
    return(S)
}
```

Appendix B

Parameter Estimation

The input for estimating the parameters is a set of games in the same format as used for creating a table (A). The output is a set of parameters for the teams involved.

```
Parameters <- function(games) {
    teams <- sort(unique(c(games[,1], games[,2])), decreasing = FALSE)
    n <- length(teams)
    g <- nrow(games)
    Y <- matrix(0,2*g,1)
    for (i in 1:g) {
        Y[((2*i)-1)] <- games[i,3]
        Y[(2*i)] <- games[i,4]
    }
    X <- matrix(0,2*g,((2*n)+1))
    for (i in 1:g) {
        M <- which(teams == games[i,1])
        N <- which(teams == games[i,2])
        X[((2*i)-1),M] <- 1
        X[((2*i)-1),N+n] <- -1
        X[(2*i),N] <- 1
        X[(2*i),M+n] <- -1
        X[((2*i)-1),((2*n)+1)] <- 1
    }
    XX <- X[,-1]
    parameters <- glm(formula = Y ~ 0 + XX, family = poisson)
    Z <- c(0, coefficients(parameters))
    P <- data.frame(row.names=teams, Attack=Z[1:n], Defence=Z[(n+1):(2*n)])
    return(list(teams=P,home=as.numeric(Z[2*n+1])))
}
```

Appendix C

Probability Table

An input of a set of parameters in the format created by the previous function (B) along with two teams that the probability table is wished to be created about. These two teams need to have "hometeam" around their names. The output is a table of probabilities for the set of results possible for a football match.

```
ProbTable <- function(parameters,hometeam,awayteam) {
    teams <- rownames(parameters$teams)
    P <- parameters$teams
    home <- parameters$home
    a <- which(teams == hometeam)
    b <- which(teams == awayteam)
    lambdaa <- exp(P[a,]$Attack - P[b,]$Defence + home)
    lambdab <- exp(P[b,]$Attack - P[a,]$Defence)
    A <- as.numeric()
    B <- as.numeric()
    for(i in 0:6) {
        A[(i+1)] <- dpois(i,lambdaa)
        B[(i+1)] <- dpois(i,lambdab)
    }
    A[8] <- 1 - sum(A[1:7])
    B[8] <- 1 - sum(B[1:7])
    name <- c("0","1","2","3","4","5","6","7+")
    zero <- mat.or.vec(8,1)
    C <- data.frame(row.names=name)
    for(j in 1:8) {
        for(k in 1:8) {
            C[j,k] <- A[k]*B[j]
        }
    }
    colnames(C) <- name
    return(round(C*100,2))
}
```

Appendix D

Result Probabilities

Inputting the probability table from the previous function (C) gives an output of three probabilities for the different outcomes of a football match.

```
ResultProbs <- function(probs) {
    R <- matrix(0,3,1)
    n <- length(probs)
    for(i in 1:n) {
        for(j in 1:n) {
            if(i > j) {
                R[3] <- R[3] + probs[i,j]
            } else {
                if(i == j) {
                    R[2] <- R[2] + probs[i,j]
                } else {
                    R[1] <- R[1] + probs[i,j]
                }
            }
        }
    }
    return(list(HomeWin=R[1], Draw=R[2], AwayWin=R[3]))
}
```

Appendix E

Simulate a Full Season

An input of the parameters from B will give an output of a set of games in the same format as shown in 4.1.

```
Games <- function(parameters) {
    teams <- rownames(parameters)
    P <- parameters$teams
    home <- parameters$home
    n <- length(teams)
    C <- data.frame()
    row <- 1
    for (i in 1:n) {
        for (j in 1:n) {
            if (i != j) {
                C[row,1] <- teams[i]
                C[row,2] <- teams[j]
                C[row,3] <- rpois(1, exp(P[i,]$Attack - P[j,]$Defence + home))
                C[row,4] <- rpois(1, exp(P[j,]$Attack - P[i,]$Defence))
                row <- row + 1
            }
        }
    }
    return(C)
}
```

Appendix F

Simulate k Seasons

A same input as the previous function but a number k inputted as well to specify how many sets of games to simulate. Tables are then created for each set of games and the output is a table giving each finishing position of each team from each table.

```
Sim <- function(parameters, k) {
    teams <- rownames(parameters$teams)
    n <- length(teams)
    A <- data.frame(row.names=teams)
    for(i in 1:k) {
        T <- Table(Games(parameters))
        for(j in 1:n) {
            A[teams[j],i] <- which(T == teams[j])
        }
    }
    return(A)
}
```

Appendix G

Simulation Statistics

Entering the set of simulations outputted from F gives statistics about each team's set of finishing positions.

```
SimStats <- function(Sim) {
    teams <- rownames(Sim)
    n <- length(teams)
    zero <- matrix(0,n,1)
    M <- data.frame(Team=teams, Average=rowMeans(Sim), StDev=zero, Mode=zero)
    for(i in 1:n) {
        a <- as.numeric(Sim[i,])
        M[i,"StDev"] <- sd(a)
        M[i,"Mode"] <- names(sort(-table(a)))[1]
    }
    N <- data.frame(row.names=c(1:20), M[with(M, order(Average)), ])
    return(N)
}
```

Appendix H

Multiple Parameter

An input of a set of parameters as outputted from B along with a number k gives an output of statistics about the k sets of new parameters that are estimated.

```
MultiPara <- function(realpara,k) {
    teams <- rownames(realpara$teams)
    n <- length(teams)
    zero <- matrix(0,n,1)
    Q <- data.frame(row.names=teams, Attack=zero, ASQ=zero, Defence=zero, DSQ=zero)
    homepara <- 0
    homeSQ <- 0
    for(i in 1:k) {
        G <- Games(realpara)
        P <- Parameters(G)
        for(j in 1:n) {
            Q[j,1] <- Q[j,1] + P$teams[j,1]
            Q[j,2] <- Q[j,2] + P$teams[j,1]^2
            Q[j,3] <- Q[j,3] + P$teams[j,2]
            Q[j,4] <- Q[j,4] + P$teams[j,2]^2
        }
        homepara <- homepara + P$home
        homeSQ <- homeSQ + P$home^2
    }
    R <- data.frame(row.names=teams,
                Attack.bias=realpara$teams[,1] - Q[,1]/k,
                Attack.sd=(k/(k-1))*(sqrt(Q[,2]/k - (Q[,1]/k)^2)),
                Defence.bias=realpara$teams[,2] - Q[,3]/k,
                Defence.sd=(k/(k-1))*(sqrt(Q[,4]/k - (Q[,3]/k)^2)))
    return(list(teams=R, home.bias=realpara$home - homepara/k,
            home.sd=(k/(k-1))*(sqrt(homeSQ/k - (homepara/k)^2))))
}
```

Appendix I

Simulate Fixtures

An adaption to E with an extra input of a set of fixtures to be simulated. An output of the most likely scores for these fixtures is given.

```
Games1 <- function(parameters,fixtures) {
    teams <- sort(unique(c(rownames(parameters$teams))), decreasing = FALSE)
    P <- parameters$teams
    home <- parameters$home
    n <- nrow(fixtures)
    A <- data.frame()
    for(i in 1:n) {
        A[i,1] <- fixtures[i,1]
        A[i,2] <- fixtures[i,2]
        x <- which(teams == fixtures[i,1])
        y <- which(teams == fixtures[i,2])
        lambdax <- exp(P[x,]$Attack - P[y,]$Defence + home)
        lambday <- exp(P[y,]$Attack - P[x,]$Defence)
        X <- as.numeric()
        Y <- as.numeric()
        for(j in 0:6) {
            X[(j+1)] <- dpois(j,lambdax)
            Y[(j+1)] <- dpois(j,lambday)
        }
        X[8] <- 1 - sum(X[1:7])
        Y[8] <- 1 - sum(Y[1:7])
        A[i,3] <- which(X == max(X)) - 1
        A[i,4] <- which(Y == max(Y)) - 1
        A[i,5] <- (max(X))*(max(Y))
    }
    return(A)
}
```

Bibliography

[1] A. Colin Cameron and Pravin K. Trivedi. *Regression analysis of count data*. Cambridge University Press, 1998.

[2] G. M. Clarke and D. Cooke. *A basic course in Statistics*. Arnold, 2004.

[3] A. C. Davison. *Statistical Models*. Cambridge University Press, 2003.

[4] Football data. `http://www.football-data.co.uk/`, 2011.

[5] J. F. C. Kingman. *Poisson Processes*. Oxford University Press, 1993.

[6] Rao Toutenburg. *Linear Models: Least Squares and Alternatives*. Springer, 1995.

[7] Neil A. Weiss. *A Course in Probability*. Pearson Education Inc., 2006.